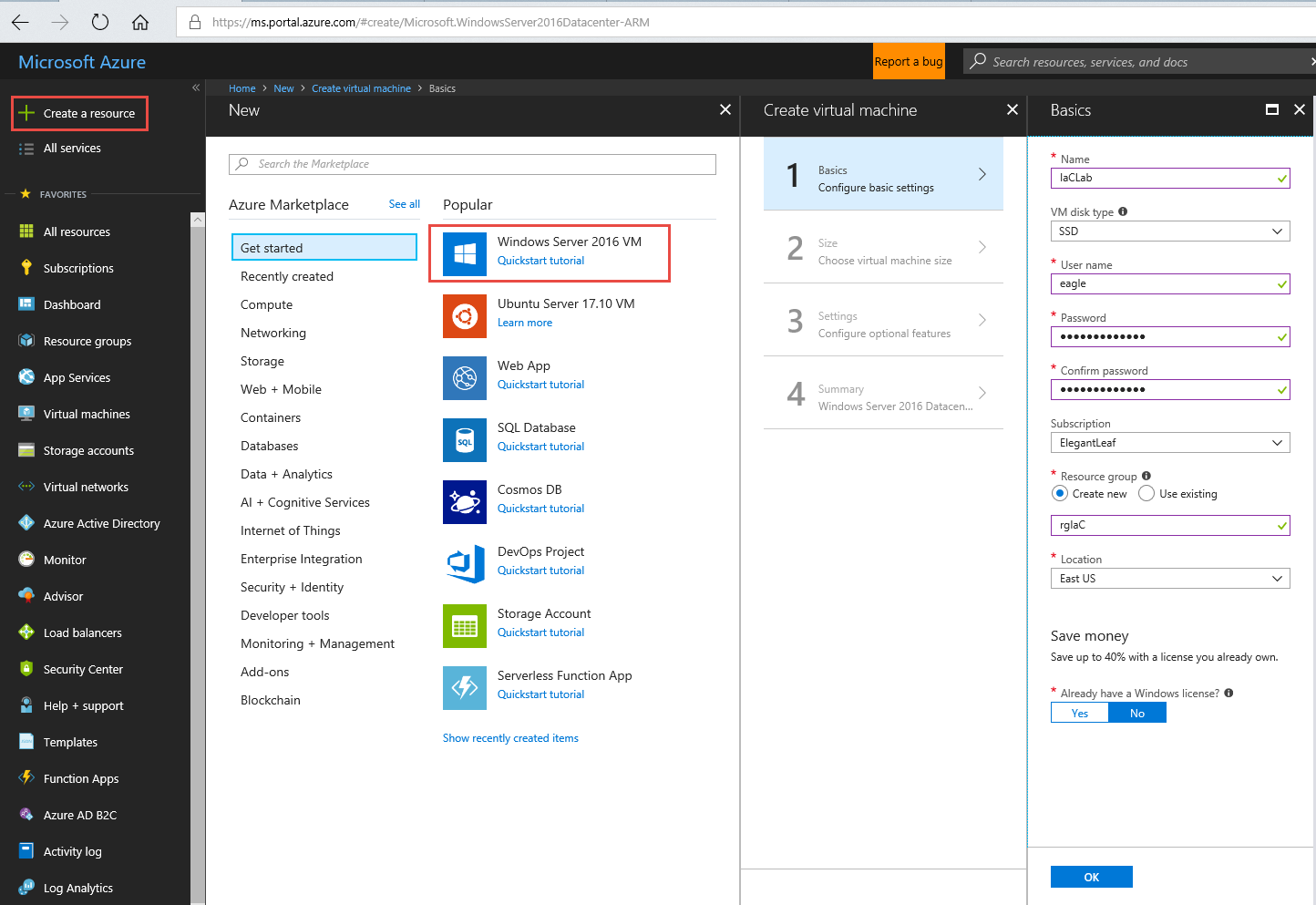
Create a new Azure dashboard, customize, clock, Service Health

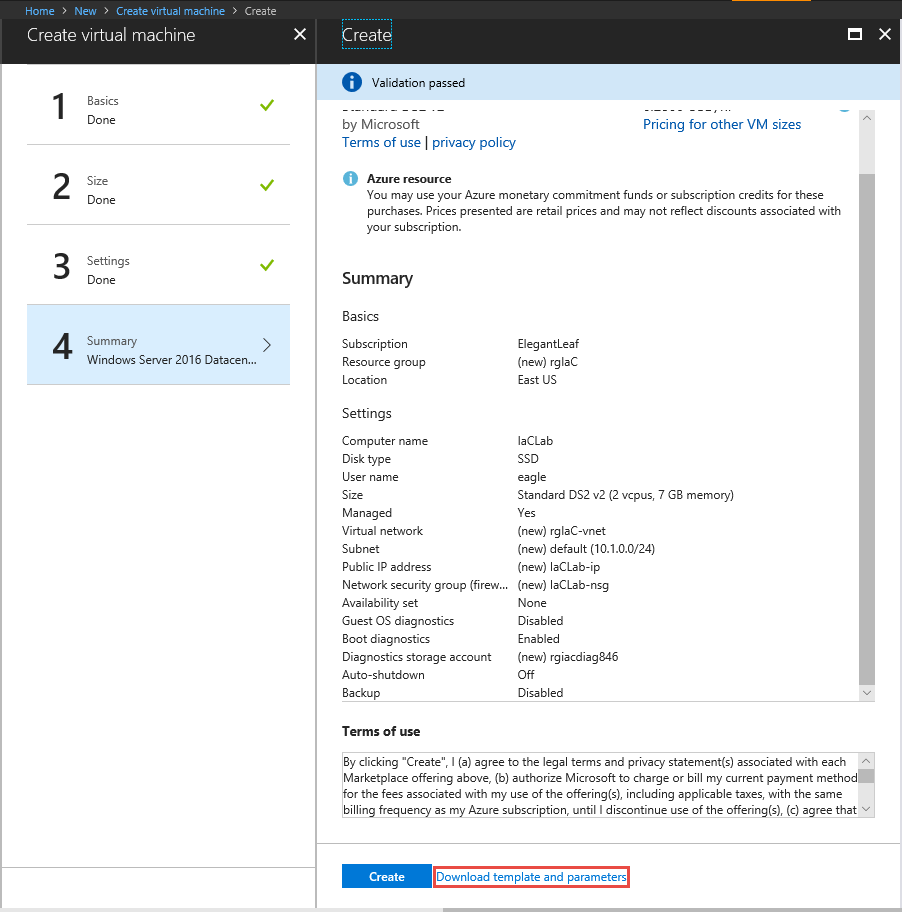
Create a Server 2016 VM for IaC work.



Size DS2\_V2: .28 per hour per vm

Blade 3: Keep defaults

Blade 4: Download template and parameters to examine

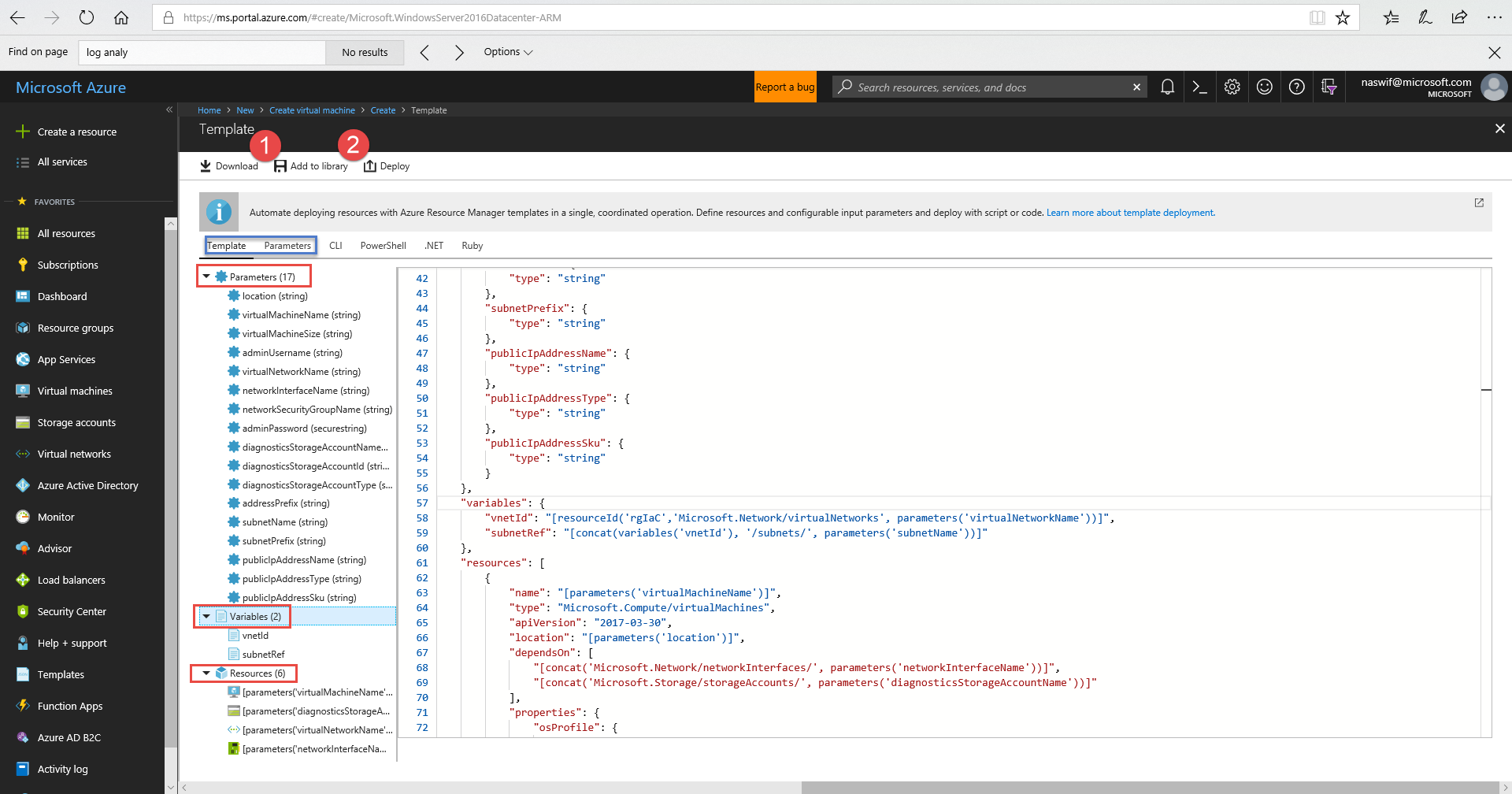


Examine The structure of a ARM template,

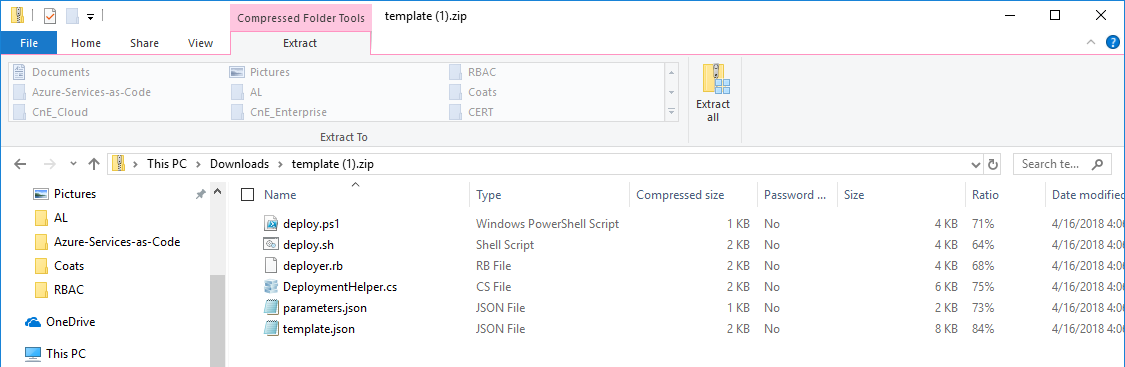
**Parameters**: Inputs for the ARM template can be inputted Programmatic or a manual response, can take strings, secure strings, integers, arrays

**Variables**: Reference an existing azure Resource like a NIC, VNET or a piece of information to be used over and over again in ARM template

**Resources**: Azure resources to deploy, with configurations of the resource.

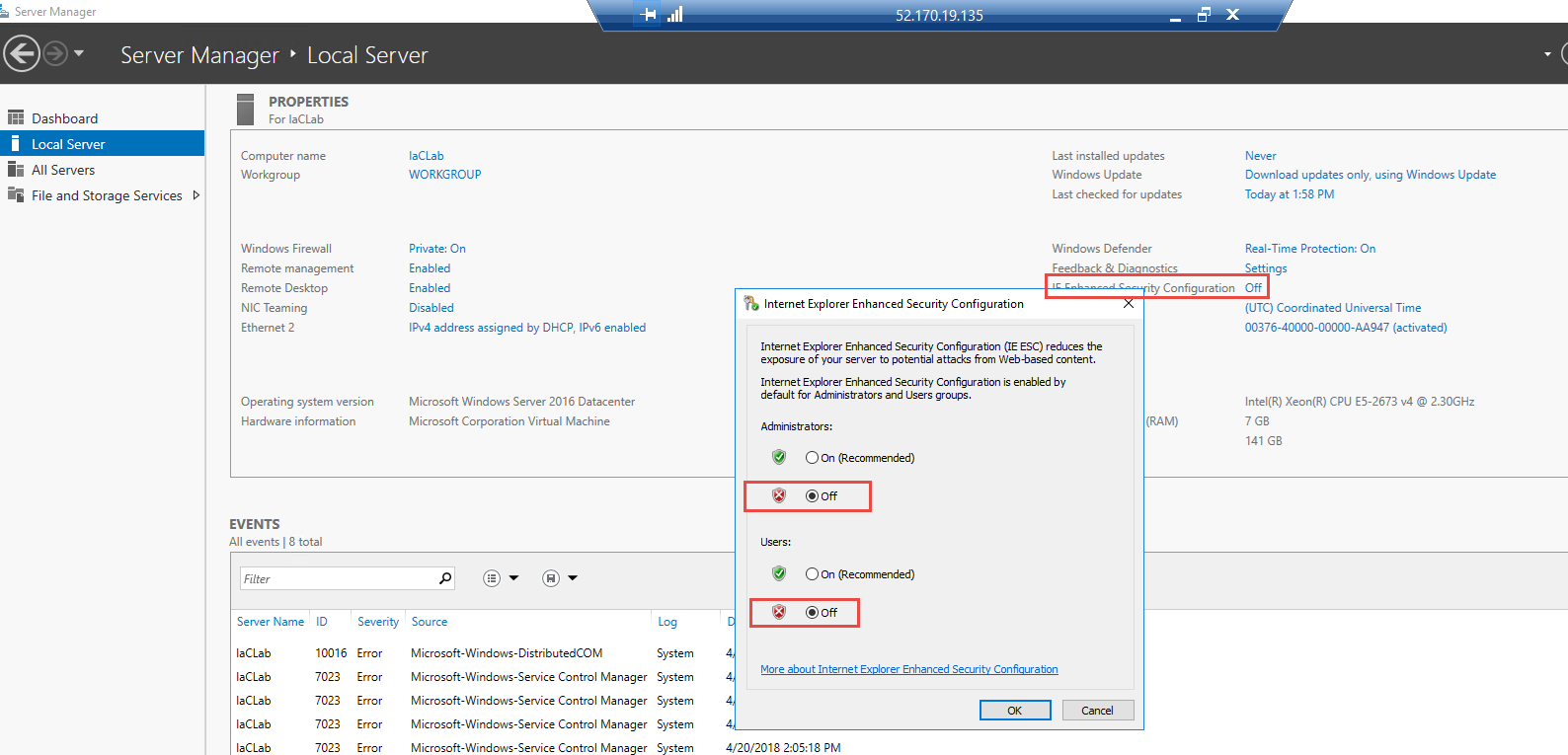


Options to Download as a .zip and Save to a My Templates Repo



Need to install some tools on IaC box:

Set enhanced security configuration off.



Launch PowerShell in Admin Mode: Install-Module AzureRM

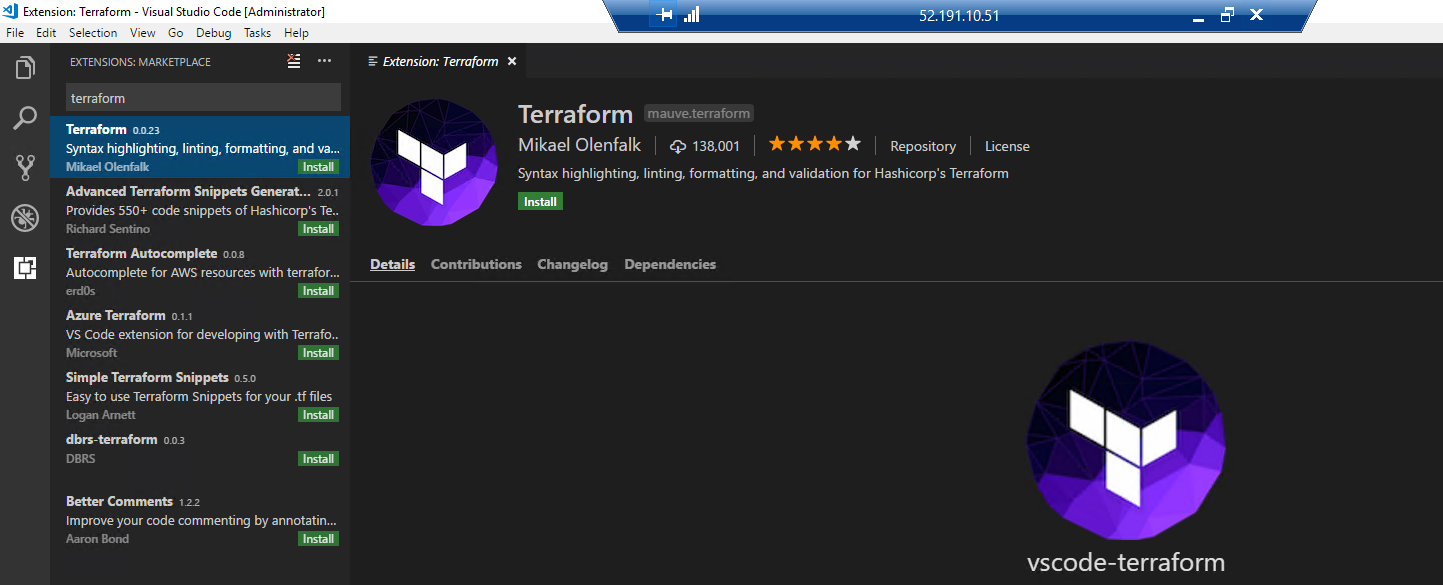
[Download and Install Azure CLI](https://aka.ms/installazurecliwindows)

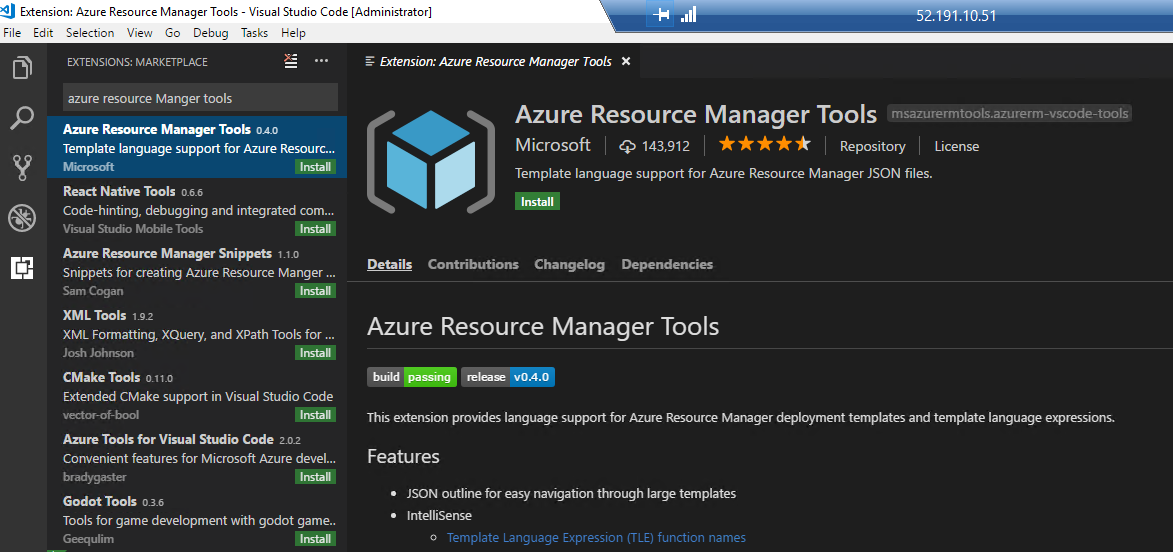
[Download and Install Visual Studio Code](https://code.visualstudio.com/docs/?dv=win)

Open Visual Studio Code

Open and configure VSC Extensions

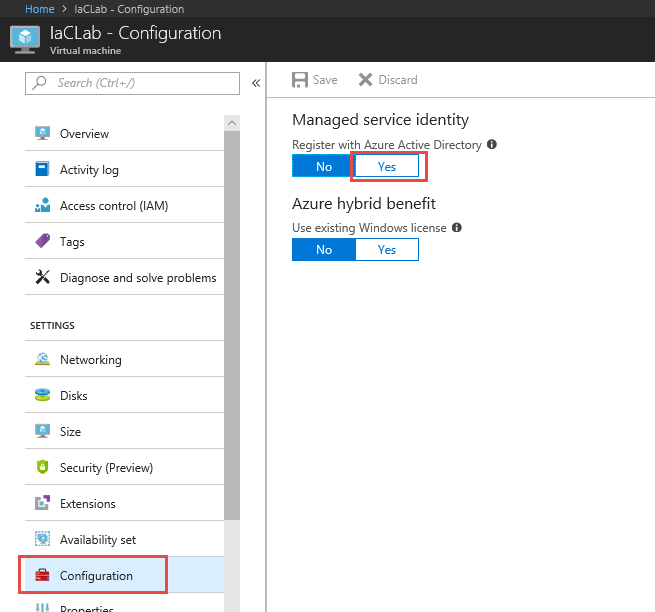
Terraform and [Azure Resource Manager Tools](https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-manager-vscode-extension)



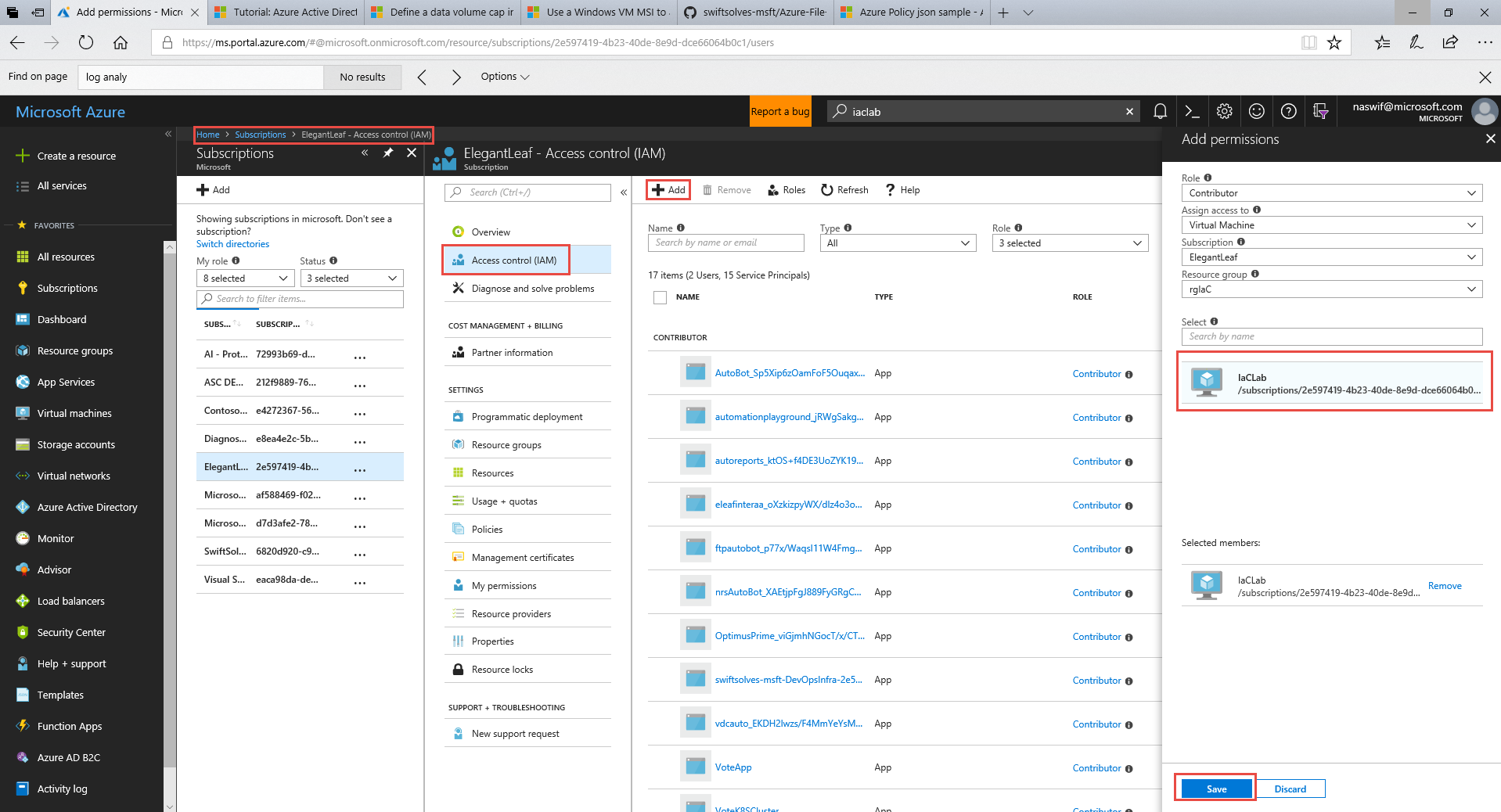


Create a Directory C:\terraform and [download terraform.exe](https://releases.hashicorp.com/terraform/0.11.7/terraform_0.11.7_windows_amd64.zip?_ga=2.31097530.444544454.1523905586-371067062.1523905586) into c:\terraform

Go to VM – Configuration and register with Azure Active Directory. This will allow PowerShell, CLI, and Terraform to authenticate and access Azure subscriptions.



Go to Subscriptions and Access control (IAM) and Add Contributor role to assign to Virtual machines, choose sub\resource group where VM is in, and choose VM to assign permissions



Register for Global VNET Peering - <https://azure.microsoft.com/en-us/updates/global-vnet-peering-region-expansion/>

Using Azure PowerShell or CloudShell – PowerShell – Signup for Global VNET peering

*# This will start registration of Global VNET Peering in Subscription, take 20 mins*

*Register-AzureRmProviderFeature ` -FeatureName AllowGlobalVnetPeering ` -ProviderNamespace Microsoft.Network*

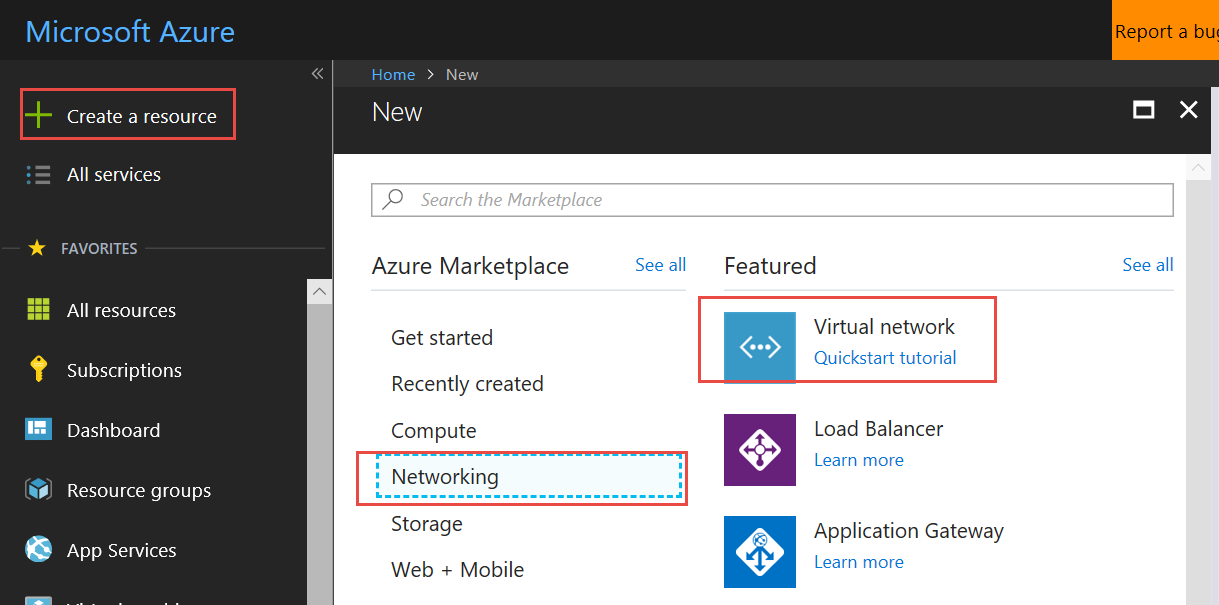
*# This will check the status of the Global VNET Peering registration*

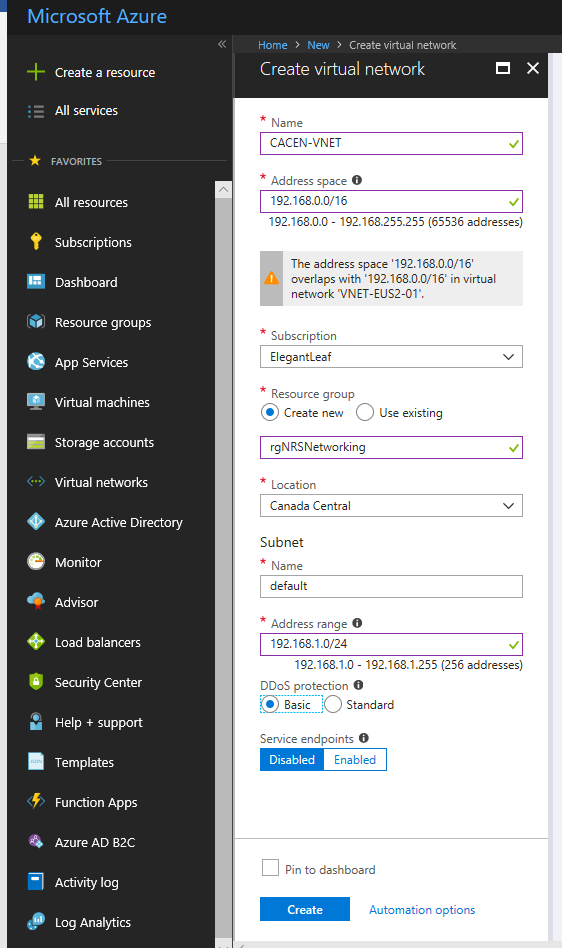
*Get-AzureRmProviderFeature ` -FeatureName AllowGlobalVnetPeering ` -ProviderNamespace Microsoft.Network*

*# This registers the overall Microsoft.Network ProviderType API*

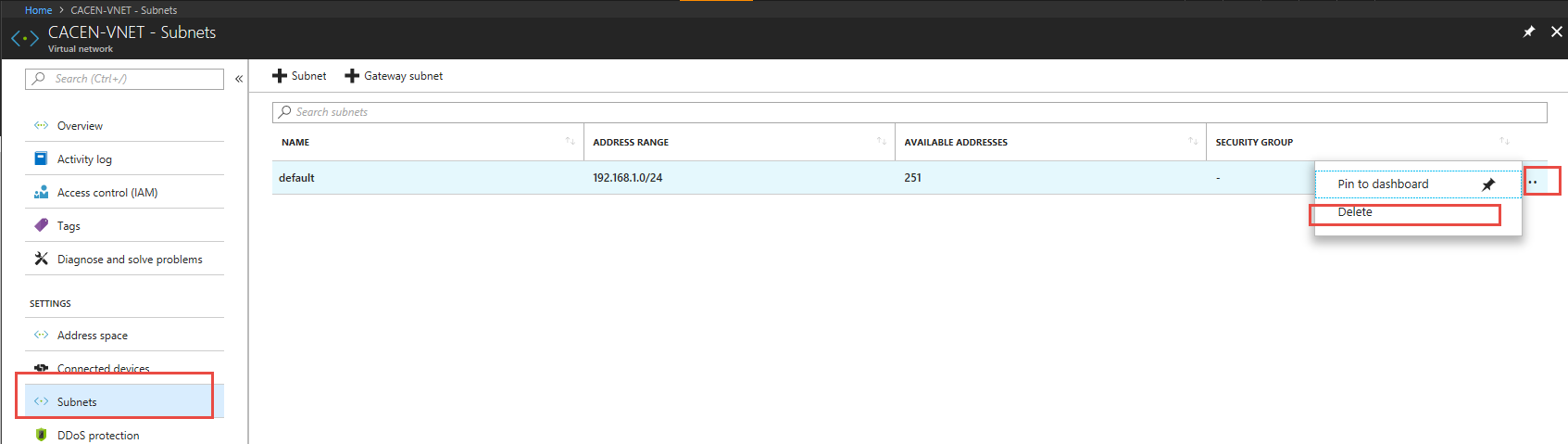
*Register-AzureRmResourceProvider ` -ProviderNamespace Microsoft.Network*

Create A Virtual Network in Canada Central – 192.168.0.0/16 – Address Space in a New Resource Group rgNetworking



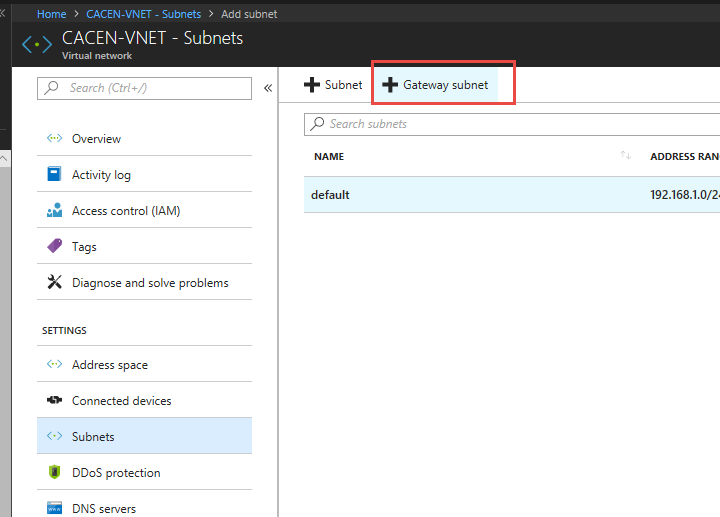


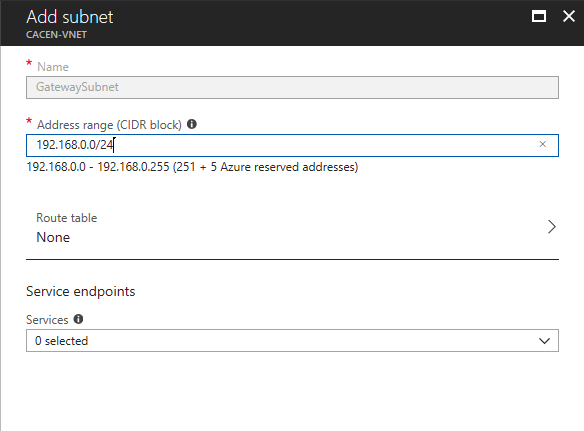
Goto Virtual network and Subnets and Delete default subnet



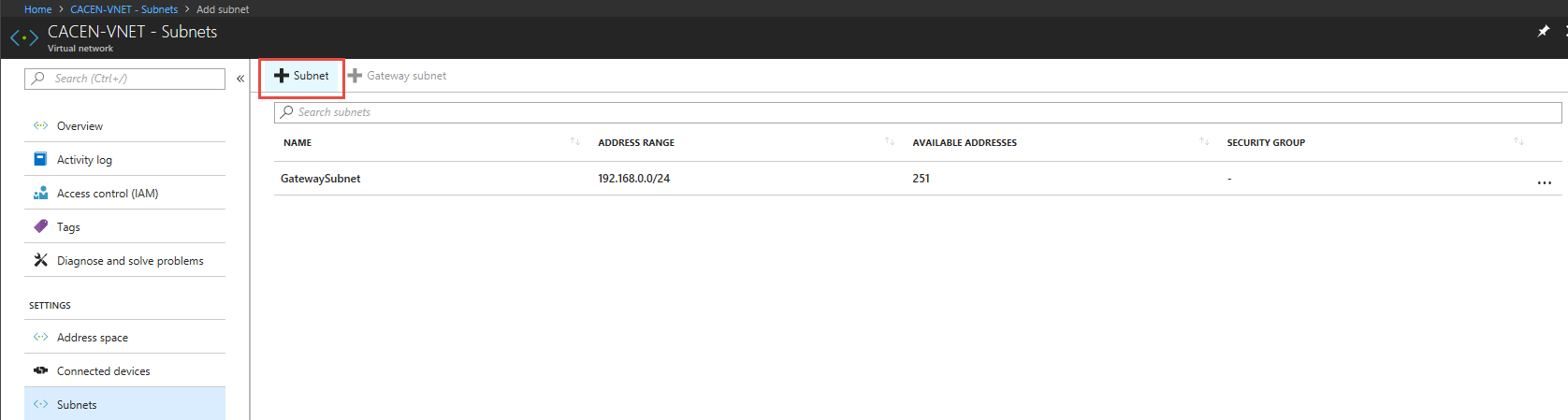
and create Default Gateway Subnet

DeafultGateway – 192.168.0.0/24

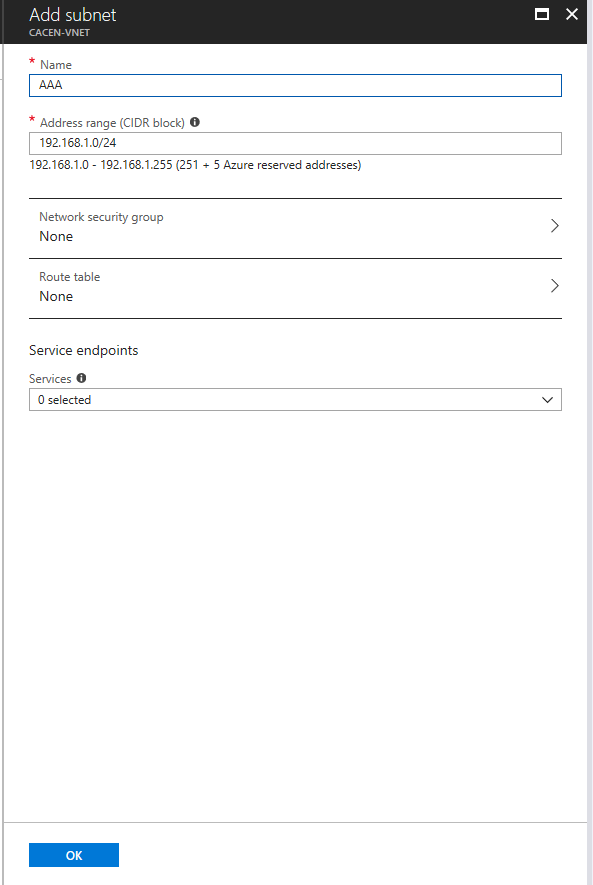




Create the following Subnets



AAA – 192.168.1.0/24

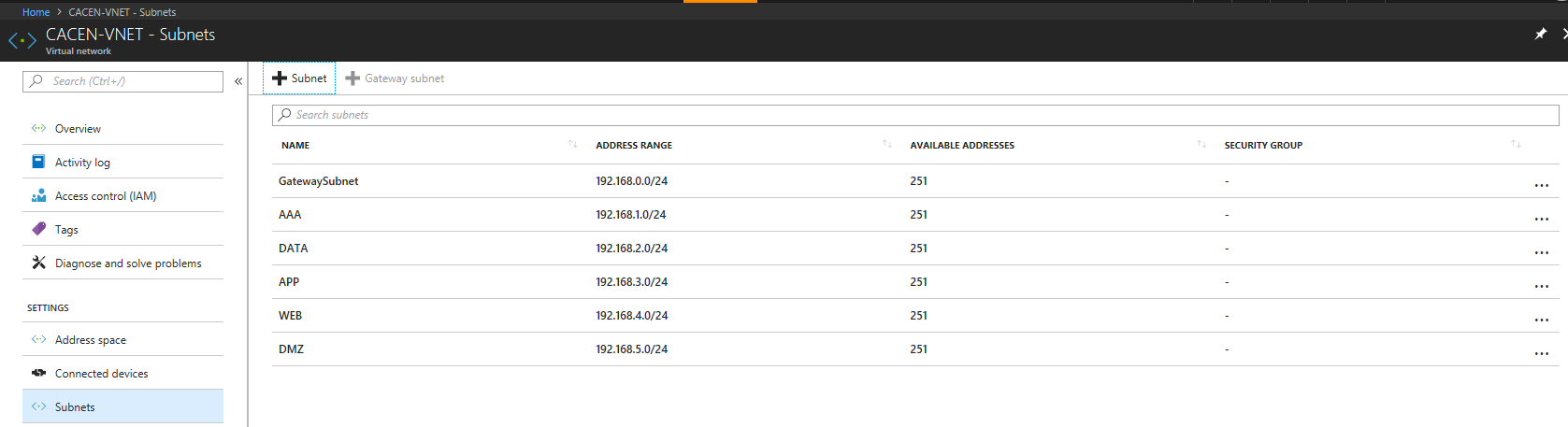


DATA – 192.168.2.0/24

APP – 192.168.3.0/24

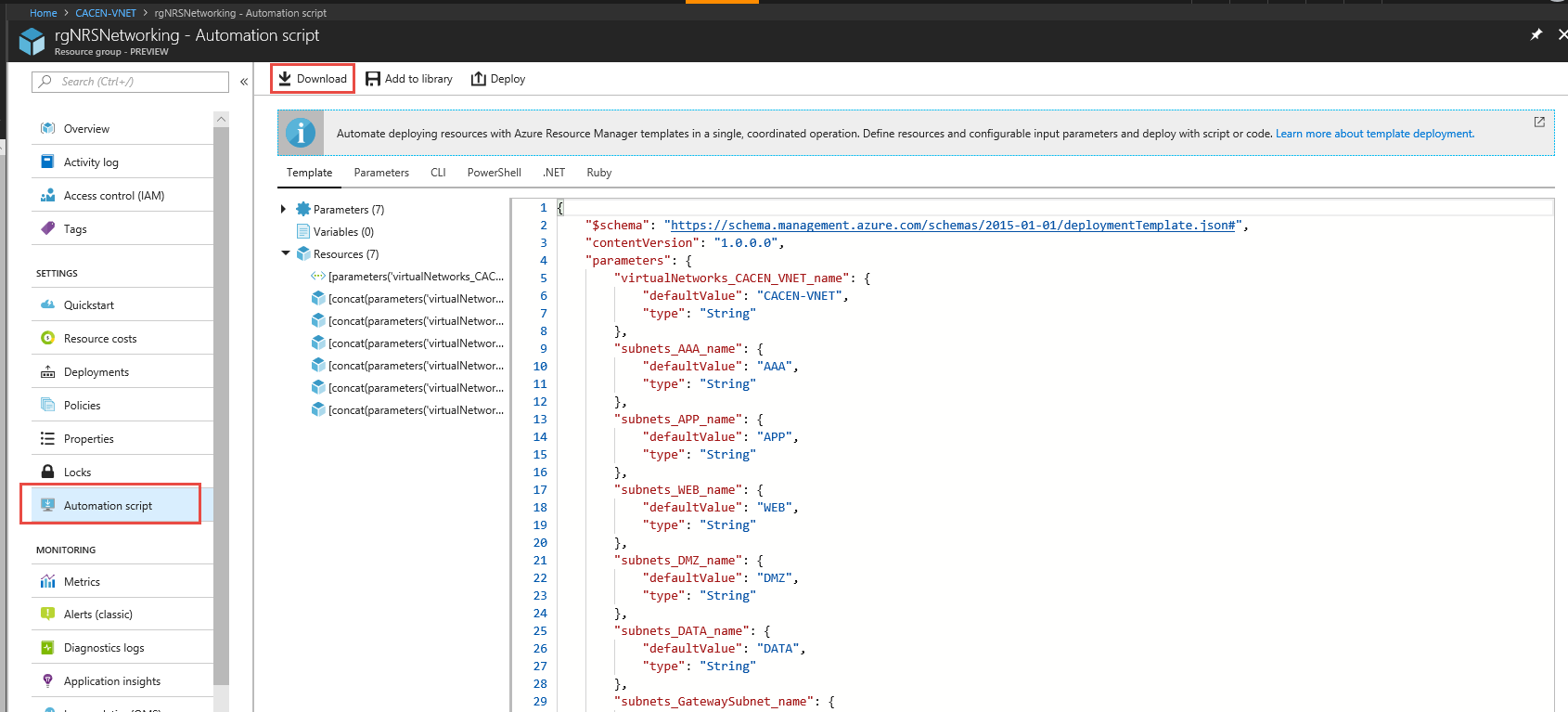
WEB – 192.168.4.0/24

DMZ – 192.168.5.0/24

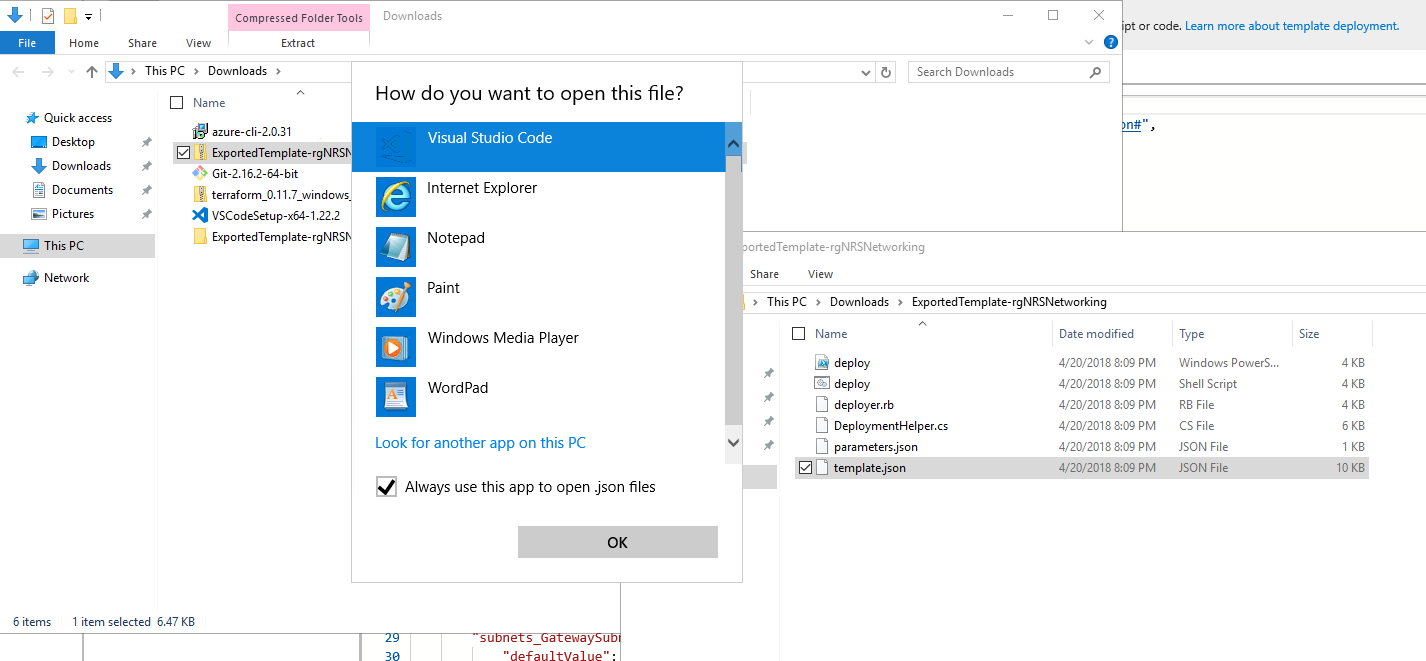


Create a Virtual network in UK South using 172.16.0.0./16 range using ARM Template

Export ARM template from rgNetworking ResourceGroup – by using Automation script and downloading



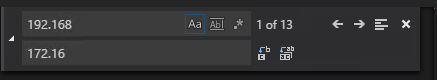
Save and UnZip File, go to folder and open template.json with Visual Studio Code



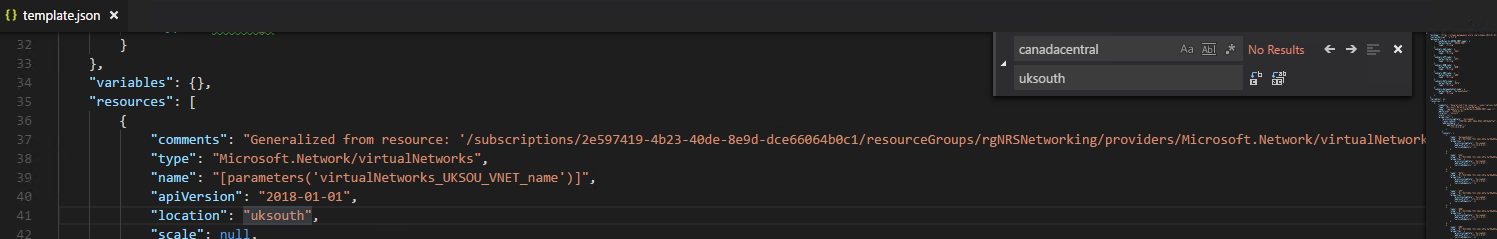
Ctrl+F Find and Replace CACEN to UKSOU



Ctrl+F Find and Replace 192.168 to 172.16

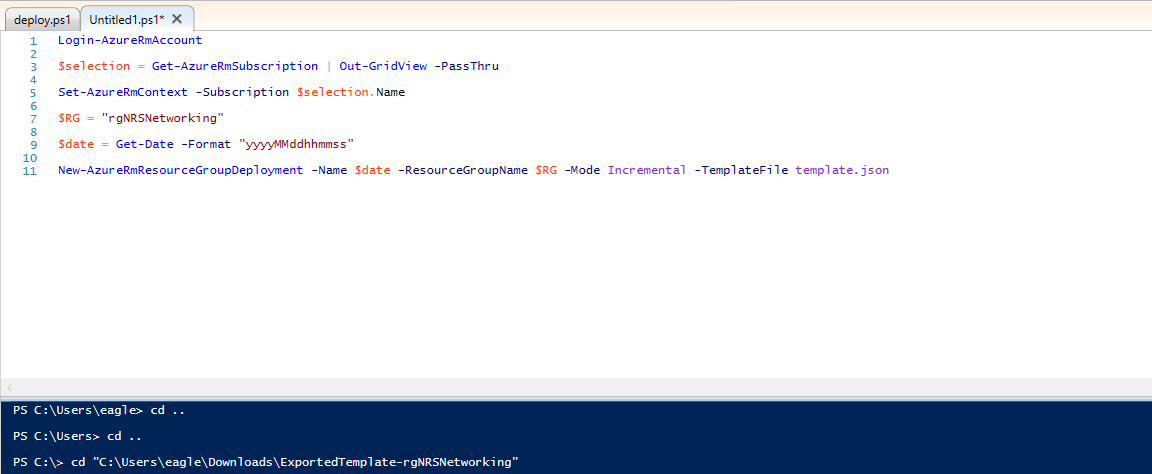


Ctrl+F Find and Replace canadacentral to uksouth



Save Template.JSON

Open PowerShell ISE from Server IaC and change directories to Template.json



*Login-AzureRmAccount*

*$selection = Get-AzureRmSubscription | Out-GridView -PassThru*

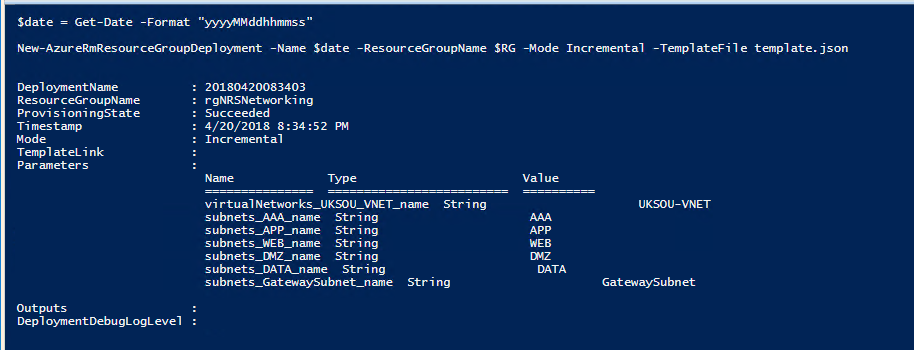
*Set-AzureRmContext -Subscription $selection.Name*

*$RG = "rgNRSNetworking"*

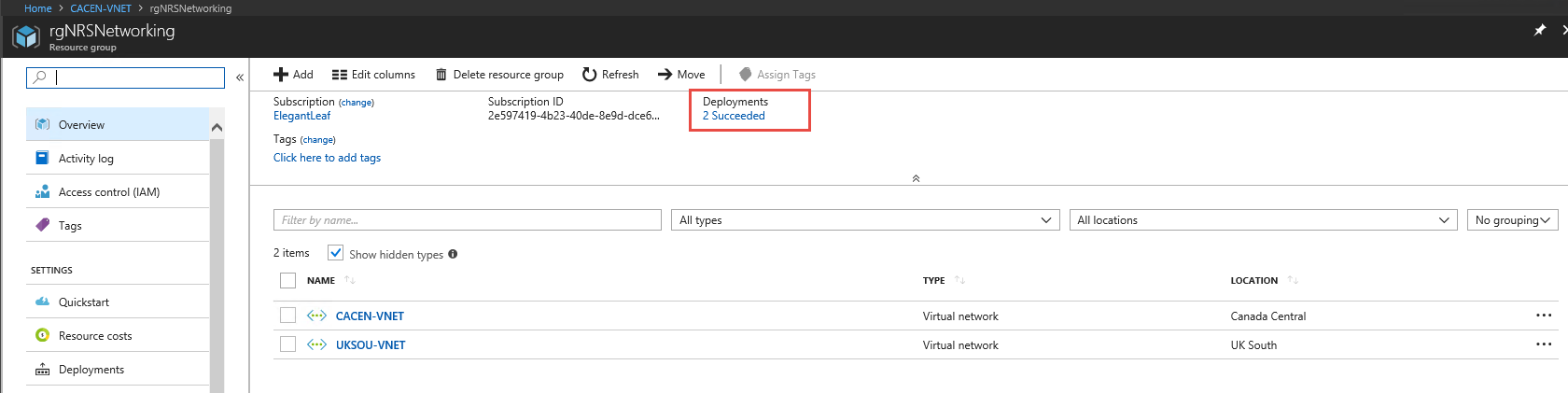
*$date = Get-Date -Format "yyyyMMddhhmmss"*

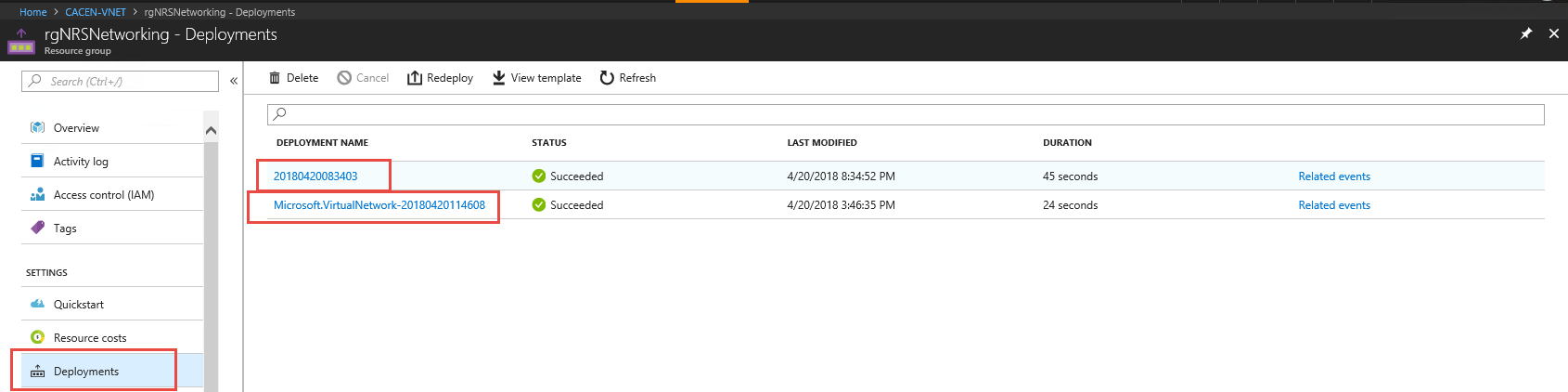
*New-AzureRmResourceGroupDeployment -Name $date -ResourceGroupName $RG -Mode Incremental -TemplateFile template.json*

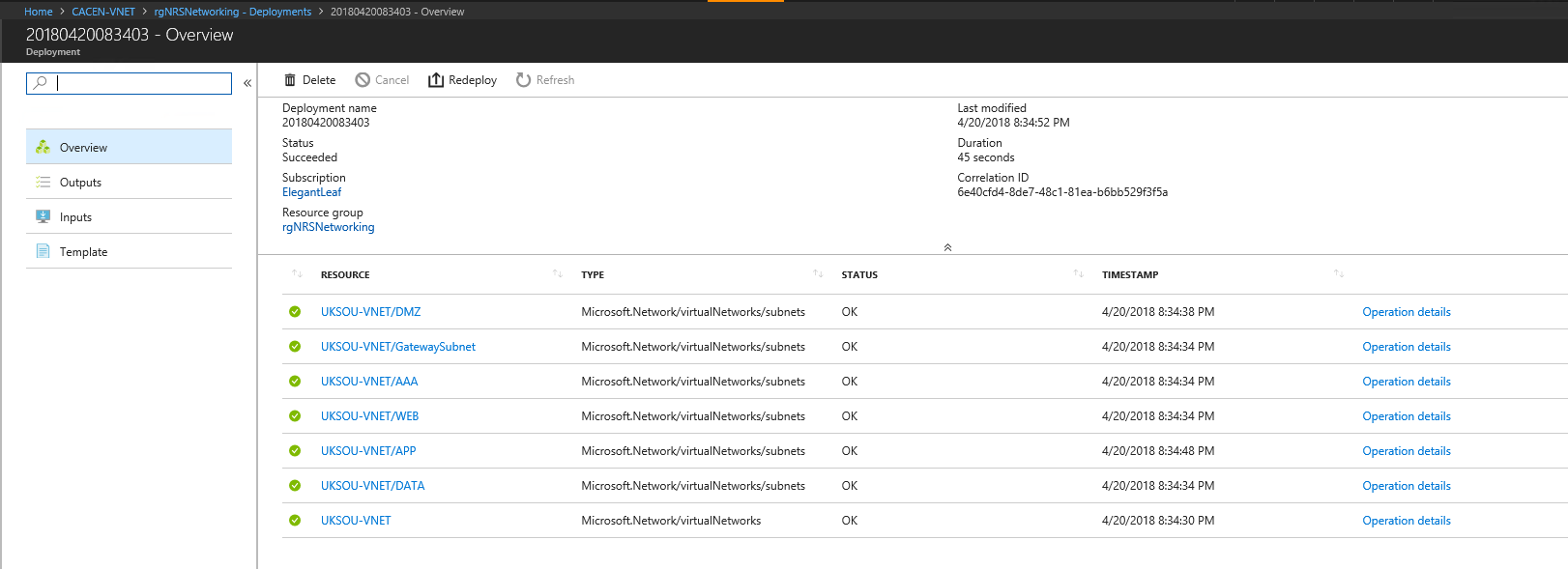
Run Powershell cmdlets which will deploy ARM Template via PS.



Goto Resource Group rgNetworking and click Deployments, examine the histories and data capture







Using PowerShell Create East US Virtual Network in 10.0.0.0/16 range

*#Login-AzureRmAccount*

*#$selection = Get-AzureRmSubscription | Out-GridView -PassThru*

*#Set-AzureRmContext -Subscription $selection.Name*

*$RG = "rgNRSNetworking"*

*$Loc = "EastUS"*

*$GWSubnet = New-AzureRmVirtualNetworkSubnetConfig -Name GatewaySubnet `*

*-AddressPrefix "10.0.0.0/24"*

*$Subnet1 = New-AzureRmVirtualNetworkSubnetConfig -Name AAA `*

*-AddressPrefix "10.0.1.0/24"*

*$Subnet2 = New-AzureRmVirtualNetworkSubnetConfig -Name DATA `*

*-AddressPrefix "10.0.2.0/24"*

*$Subnet3 = New-AzureRmVirtualNetworkSubnetConfig -Name APP `*

*-AddressPrefix "10.0.3.0/24"*

*$Subnet4 = New-AzureRmVirtualNetworkSubnetConfig -Name WEB `*

*-AddressPrefix "10.0.4.0/24"*

*$Subnet5 = New-AzureRmVirtualNetworkSubnetConfig -Name DMZ `*

*-AddressPrefix "10.0.5.0/24"*

*New-AzureRmVirtualNetwork -Name EUS-VNET -ResourceGroupName $RG `*

*-Location $Loc -AddressPrefix "10.0.0.0/16" -Subnet $GWSubnet,$Subnet1,$Subnet2,$Subnet3,$Subnet4,$Subnet5*

Reviewing the Resource Group Deployments, notice that the deployment of East US does not show up in deployment history. Azure PowerShell calls Azure Resource Manager REST APIs but does not interact with ARM engine like ARM templates or GUI Portal Create Resource.

[Terraform Ex.](https://www.terraform.io/docs/providers/azurerm/index.html) On IaCLab VM open Visual Studio Code and create a new file and save as type terraform .tf – WUS-VNET.tf

We need to indicate that Terraform will use a provider type so it know which APIs from which vendor to use and pull down. In the file add:

# Configure the Azure Provider

provider "azurerm" { }

Next we need to refer to the existing Resource Group – rgNetworking

# Create/refer a resource group

resource "azurerm\_resource\_group" "rgNetworking" {

name **=** "rgNetworking"

location **=** "Canada Central"

}

Next we will need to add the Resource Script Blocks for a Virtual Network

# Create a virtual network within the resource group

resource "azurerm\_virtual\_network" "WUS-VNET" {

name **=** "WUS-VNET"

address\_space **=** ["192.170.0.0/16"]

location **=** "${azurerm\_resource\_group.network.location}"

resource\_group\_name **=** "${azurerm\_resource\_group.network.name}"

subnet {

name **=** "GatewaySubnet"

address\_prefix **=** "192.170.0.0/24"

}

subnet {

name **=** "AAA"

address\_prefix **=** "192.170.1.0/24"

}

subnet {

name **=** "DATA"

address\_prefix **=** "192.170.2.0/24"

}

subnet {

name **=** "APP"

address\_prefix **=** "192.170.3.0/24"

}

subnet {

name **=** "WEB"

address\_prefix **=** "192.170.4.0/24"

}

subnet {

name **=** "DMZ"

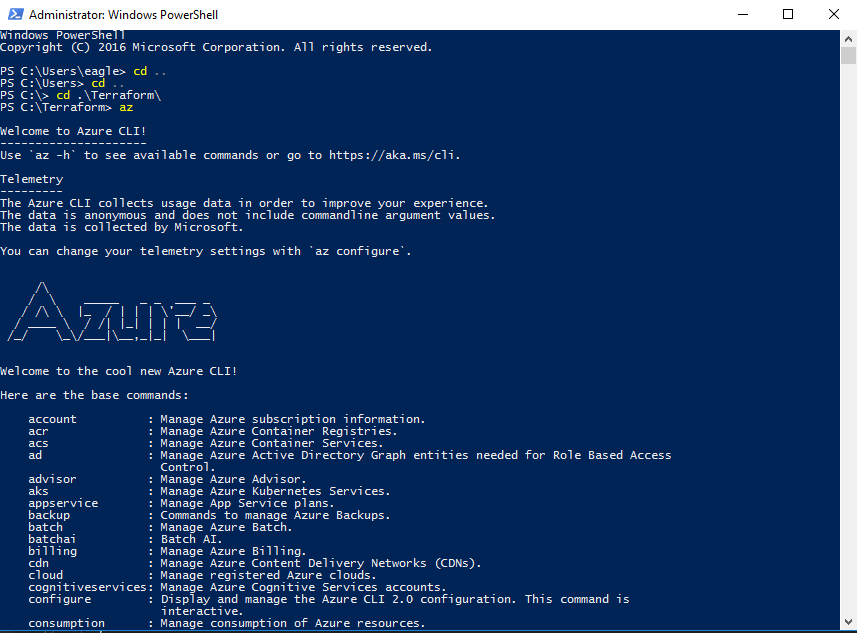
address\_prefix **=** "192.170.5.0/24"

}

}

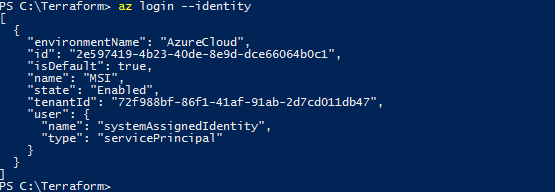
It should look like this:

Open PowerShell Change Directory to C:\Terraform and Launch Azure CLI by using *az*

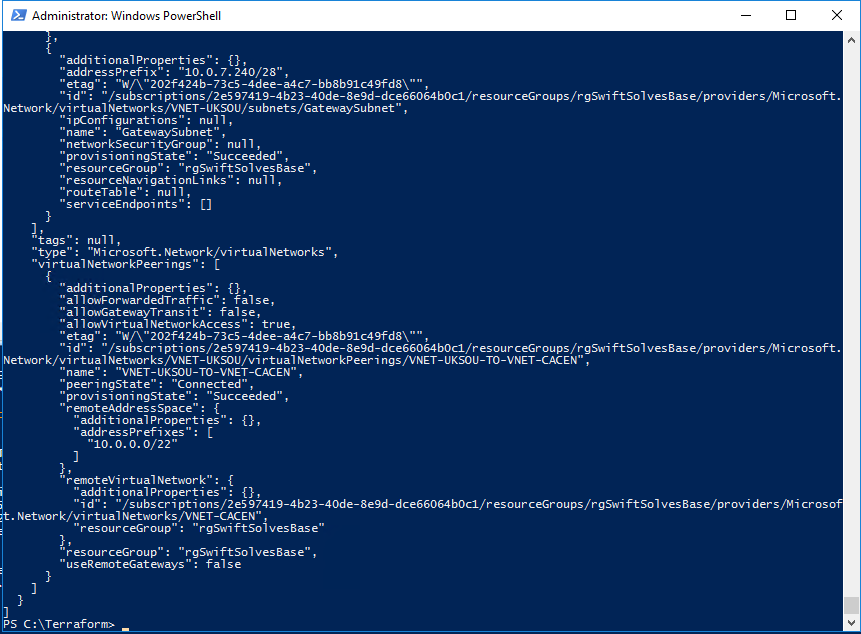


[Log into Azure CLI using the MSI – VM Extension](https://docs.microsoft.com/en-us/azure/active-directory/managed-service-identity/how-to-use-vm-sign-in#azure-cli) the server is the identity not a user, this way we don’t have to store user creds or secure user creds in a system like KeyVault

az login –identity



To test if authenticated and we can access run: az network vnet list

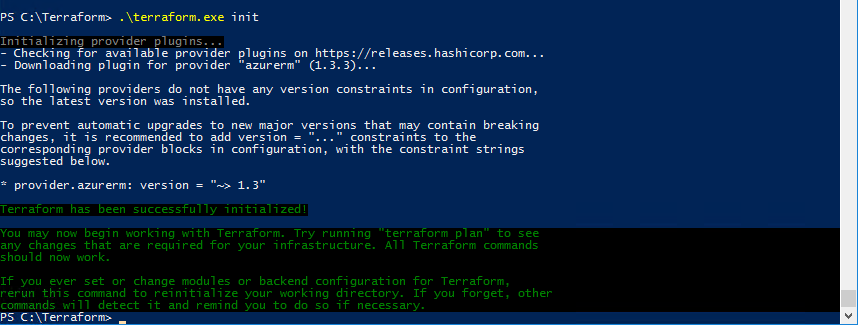


Now we need to initialize and download our terraform provider types like AzureRM based on our terraform .tf files and their subsequent resource blocks

./terraform.exe -h

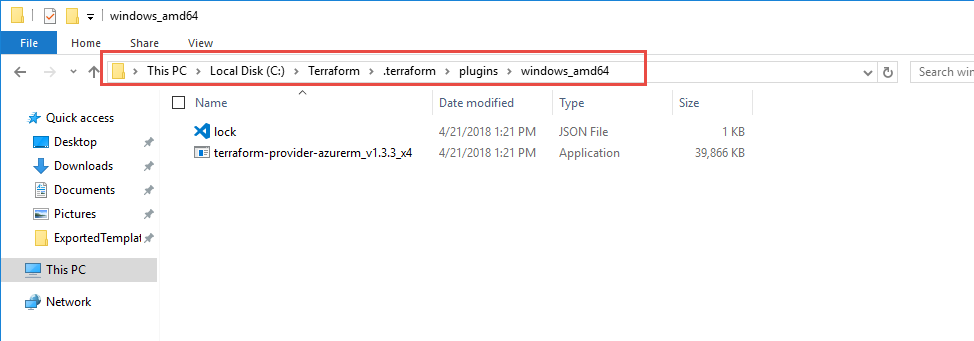


./terraform.exe init



We use this command to pull down the latest bits for Azure Resource Manager REST APIs Terraform.exe will interact with, be sure to run this over time as new APIs and versions are released for Terraform and ARM.

You will see a new folder where the bits were downloaded so terraform knows how to interact with Azure Resource Manager



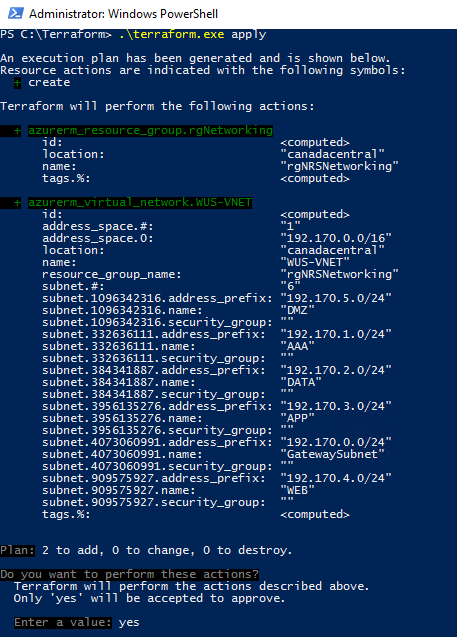
./terraform.exe plan

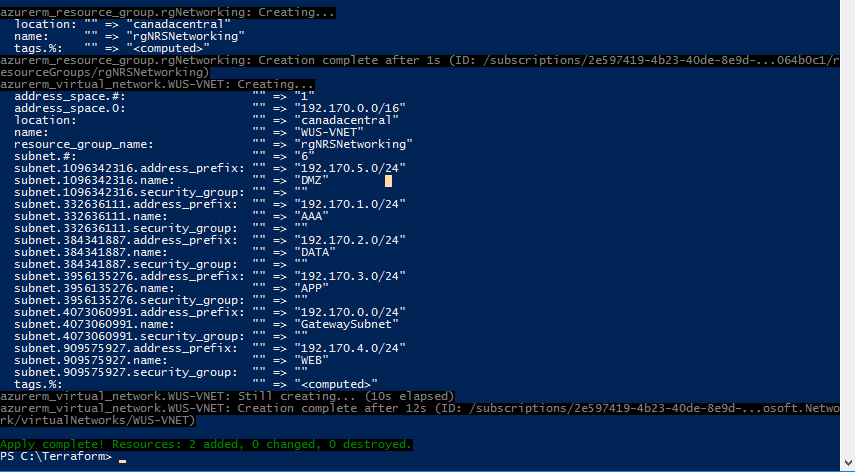
This will now run through a what if scenario of deployment showing Add/Deletes/Changes



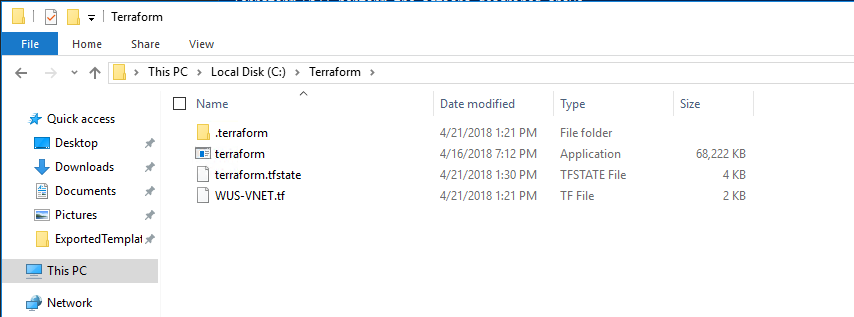
./terraform.exe apply

You will need to enter yes to confirm changes





Examine Resource Group rgNetworking again and notice the newly added VNET however Terraform natively doesn’t interact with ARM engine and does not store history in Azure just like PowerShell, it stores history locally to the client it was executed on checkout the C:\terraform folder again

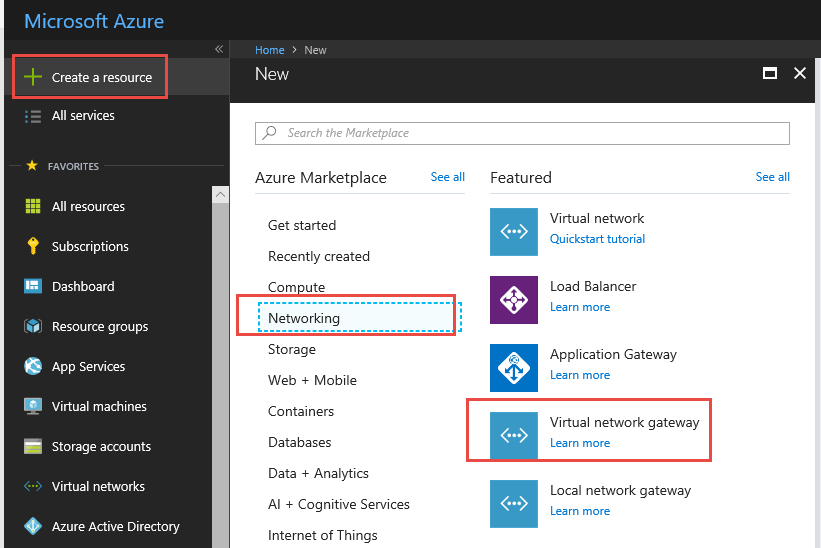


We now have a .tfstate file, after something additionally gets created again we will have a .tfstatebackup file as well and an updated .tfstate, this allows for ease of rollback

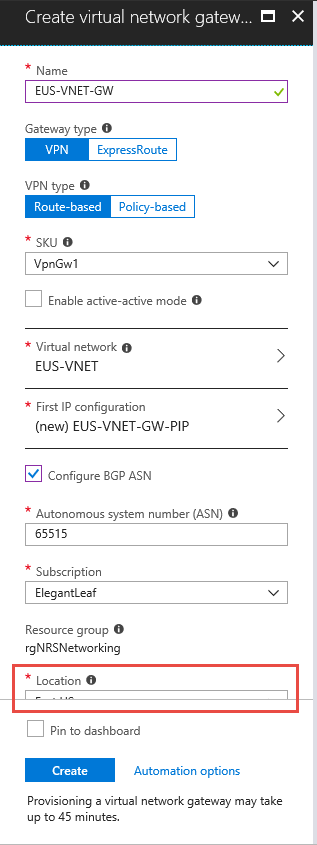
! Caution only one tfstatebackup file exists so as continue to layer and add \ change the .tfstatebackup file becomes overwritten with last previous changes.

Lets get these networks connected to one another.

Lets establish a VPN Gateway in EUS-VNET and CACEN-VNET

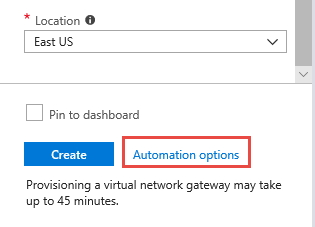


Update the Location to show East US in picker below



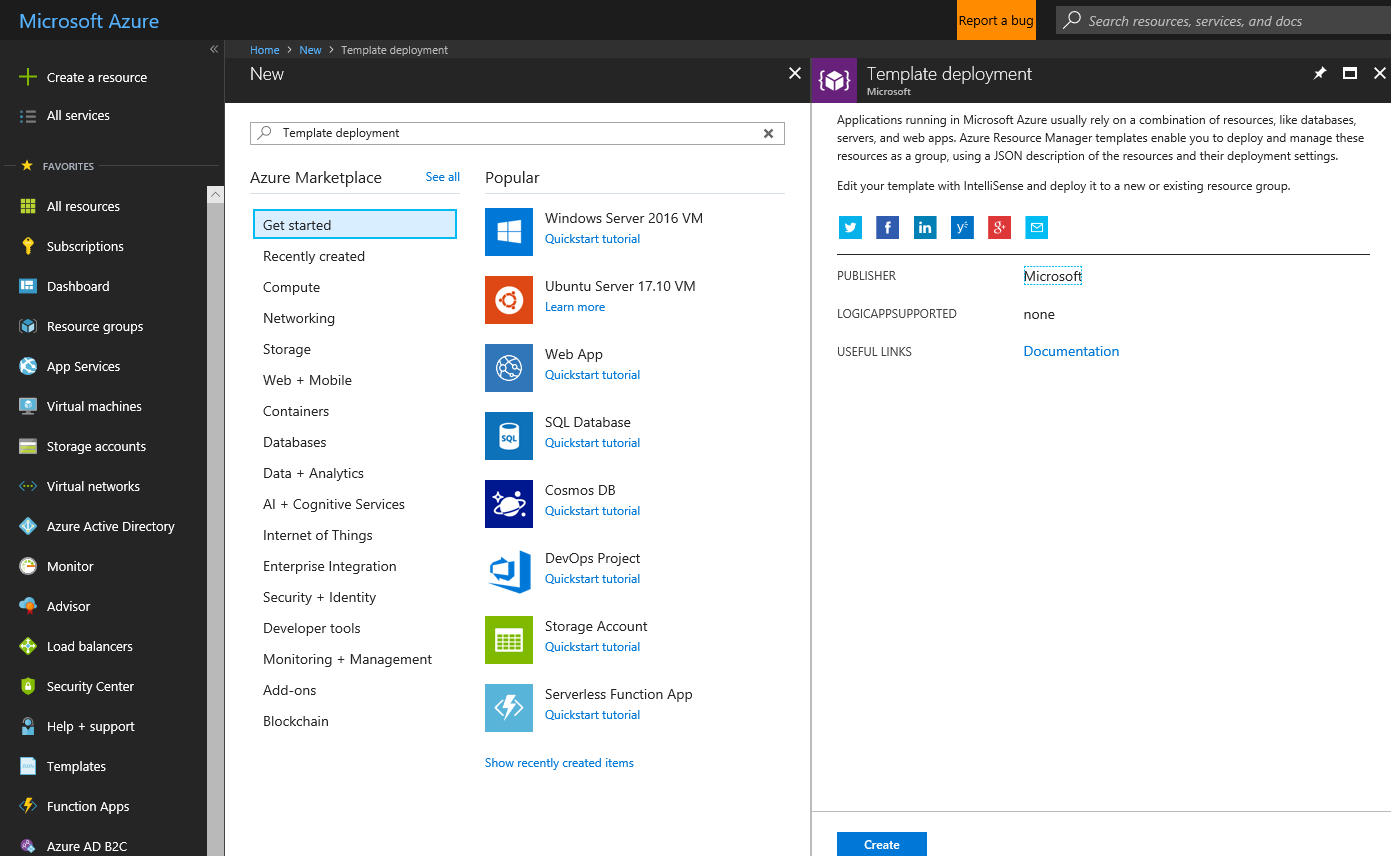
Fill out information selecting the EUS-VNET and Creating a Public IP, configure BGP ASN

!! Before creating click Automation options

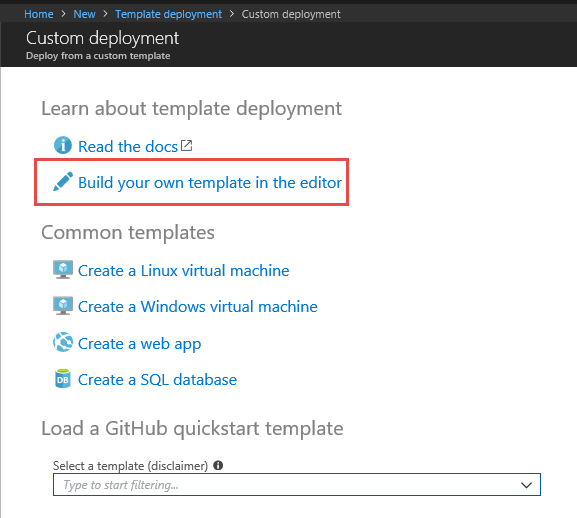


Be sure to Download as templateGW.Zip and add to library

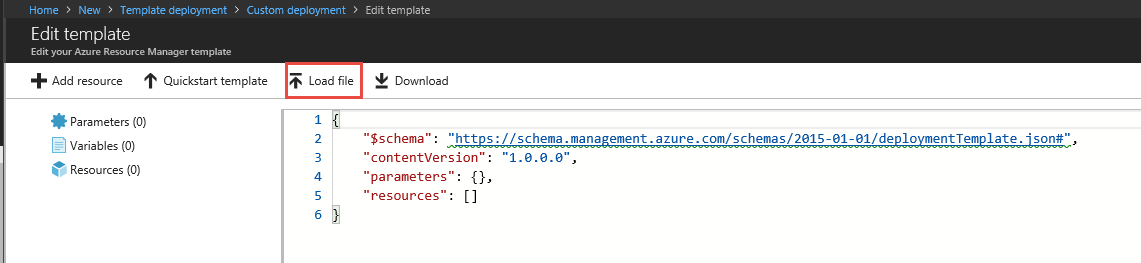
Create a Resource + and search for Template Deployment



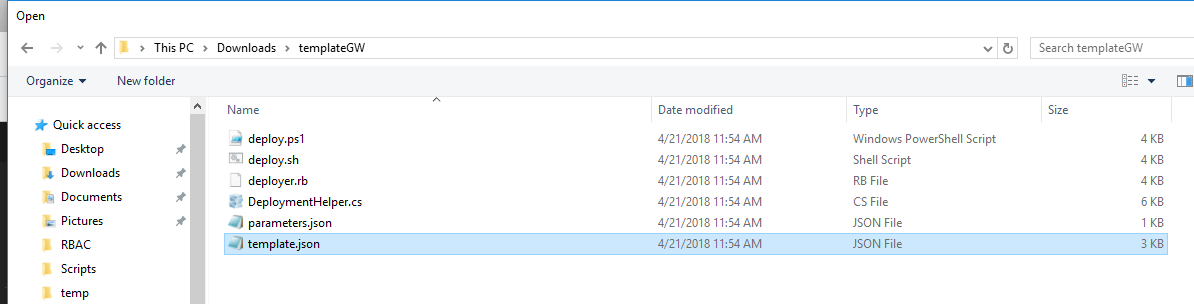
Build your own template in the editor



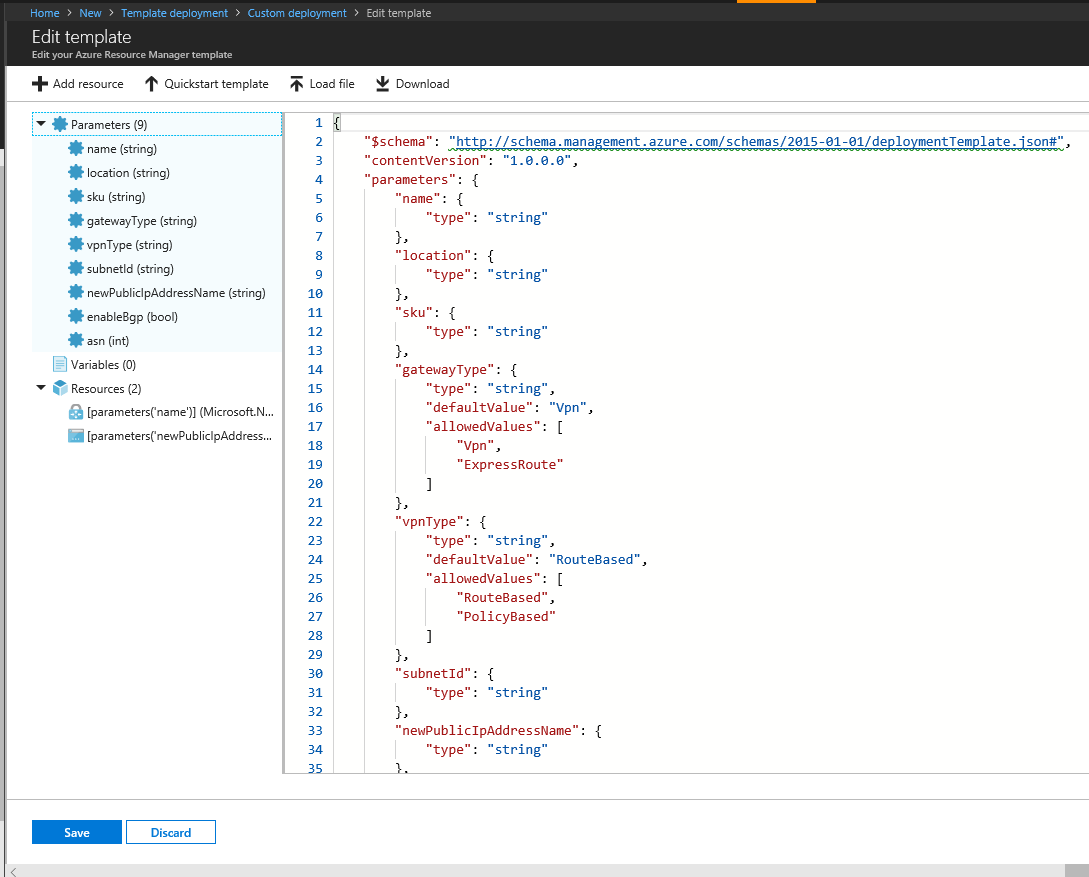
Load File



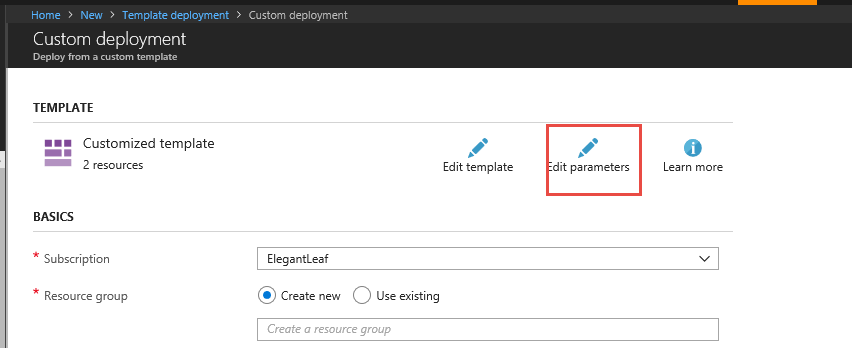
Load the templateGW\template.json file



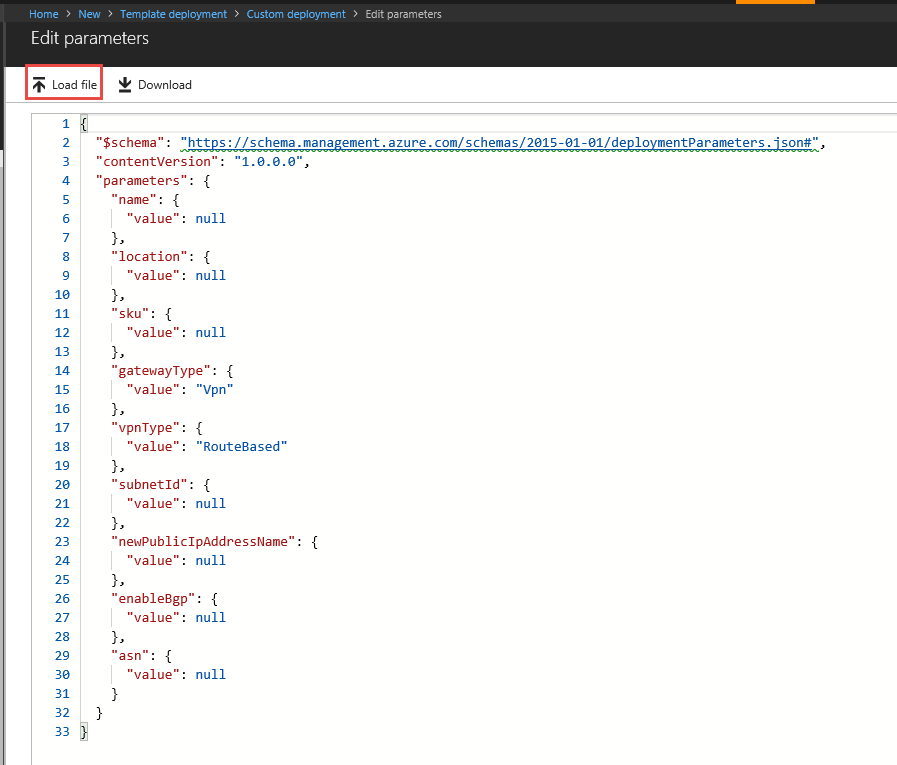
Once loaded Click Save



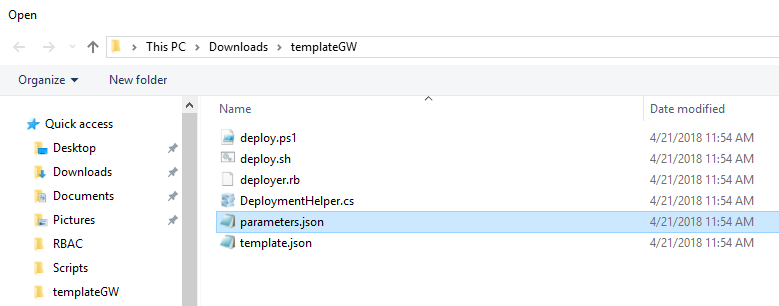
Edit parameters File



Load file



Load the parameters.json file



Once loaded make changes as we will be deploying the GW in Canada Central and it’s VNET and save:

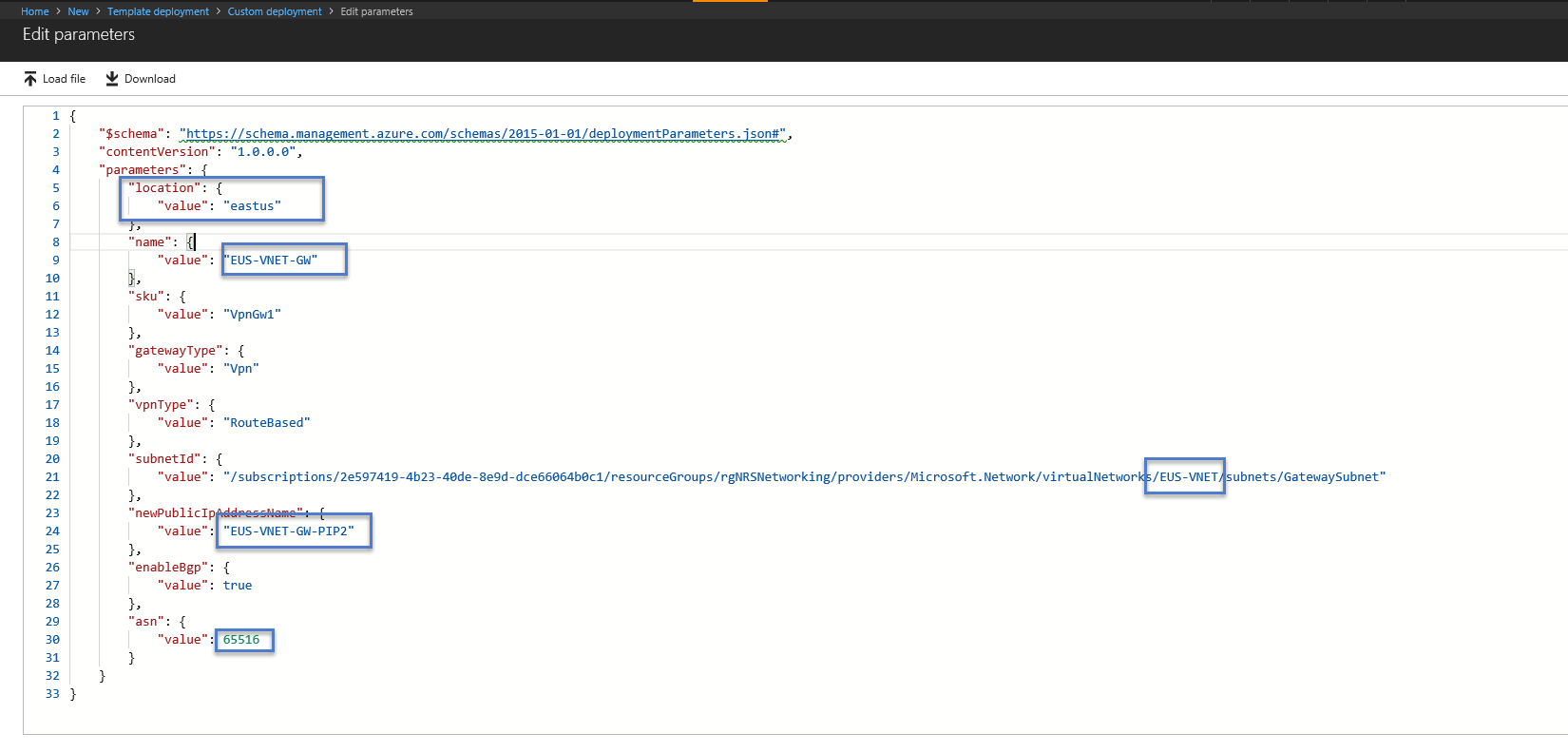
Eastus=canadacentral

EUS-VNET-GW=CACEN-VNET-GW

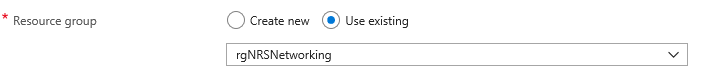
EUS-VNET=CACEN-VNET

EUS-VNET-GW-PIP=CACEN-GW-PIP

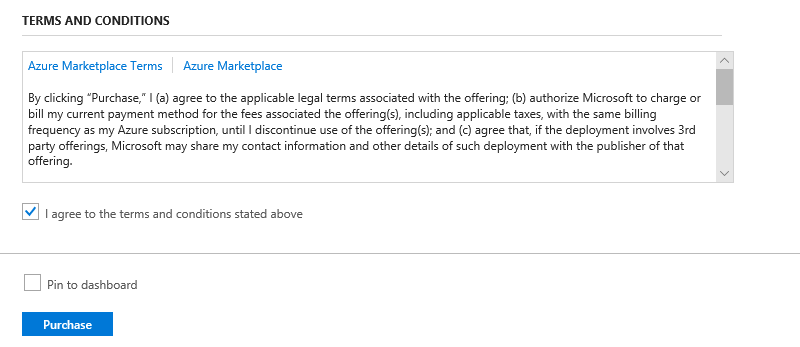
65516=65516



Save and template screen should have values answered now, be sure to set Use Existing Resource Group to rgNetworking



Scroll to bottom and accept terms and Purchase



Gateways take 20-45 Minutes to create so we can take a short 15 minute break.

Let’s Create a 2 VMs with IIS Installed in a AV Set load balanced via Azure PowerShell let’s record the time it takes to create and let’s create them in Canada Central. Other Azure VM creations via PS can be found here: <https://docs.microsoft.com/en-us/azure/virtual-machines/windows/powershell-samples>

$cred = Get-Credential

$start = Get-Date

# Variables for common values

$rgName='rgWEBVMCACEN'

$location='canadacentral'

# Create a resource group.

New-AzureRmResourceGroup -Name $rgName -Location $location

# Get existing virtual network.

$vnet = Get-AzureRmVirtualNetwork -ResourceGroupName "rgNetworking" -Name 'CACEN-VNET'

# Create a public IP address.

$publicIp = New-AzureRmPublicIpAddress -ResourceGroupName $rgName -Name 'EL4LB-PIP' `

-Location $location -AllocationMethod Dynamic

# Create a front-end IP configuration for the website.

$feip = New-AzureRmLoadBalancerFrontendIpConfig -Name 'FrontEndPool-WEB' -PublicIpAddress $publicIp

# Create the back-end address pool.

$bepool = New-AzureRmLoadBalancerBackendAddressPoolConfig -Name 'BackEndPool-WEB'

# Creates a load balancer probe on port 80.

$probe = New-AzureRmLoadBalancerProbeConfig -Name 'HealthProbe80' -Protocol Http -Port 80 `

-RequestPath / -IntervalInSeconds 360 -ProbeCount 5

# Creates a load balancer rule for port 80.

$rule = New-AzureRmLoadBalancerRuleConfig -Name 'LoadBalancerRuleWeb' -Protocol Tcp `

-Probe $probe -FrontendPort 80 -BackendPort 80 `

-FrontendIpConfiguration $feip -BackendAddressPool $bePool

# Create two NAT rules for port 3389.

$natrule1 = New-AzureRmLoadBalancerInboundNatRuleConfig -Name 'LoadBalancerRDP1' -FrontendIpConfiguration $feip `

-Protocol tcp -FrontendPort 4221 -BackendPort 3389

$natrule2 = New-AzureRmLoadBalancerInboundNatRuleConfig -Name 'LoadBalancerRDP2' -FrontendIpConfiguration $feip `

-Protocol tcp -FrontendPort 4222 -BackendPort 3389

# Create a load balancer.

$lb = New-AzureRmLoadBalancer -ResourceGroupName $rgName -Name 'WEB-EL4LB' -Location $location `

-FrontendIpConfiguration $feip -BackendAddressPool $bepool `

-Probe $probe -LoadBalancingRule $rule -InboundNatRule $natrule1,$natrule2

# Create a network security group rule for port 3389.

$rule1 = New-AzureRmNetworkSecurityRuleConfig -Name 'NetworkSecurityGroupRuleRDP' -Description 'Allow RDP' `

-Access Allow -Protocol Tcp -Direction Inbound -Priority 1000 `

-SourceAddressPrefix Internet -SourcePortRange \* `

-DestinationAddressPrefix \* -DestinationPortRange 3389

# Create a network security group rule for port 80.

$rule2 = New-AzureRmNetworkSecurityRuleConfig -Name 'NetworkSecurityGroupRuleHTTP' -Description 'Allow HTTP' `

-Access Allow -Protocol Tcp -Direction Inbound -Priority 2000 `

-SourceAddressPrefix Internet -SourcePortRange \* `

-DestinationAddressPrefix \* -DestinationPortRange 80

# Create a network security group

$nsg = New-AzureRmNetworkSecurityGroup -ResourceGroupName $RgName -Location $location `

-Name 'NetworkSecurityGroupWEBCACEN' -SecurityRules $rule1,$rule2

# Create two virtual network cards and associate with public IP address and NSG.

$nicVM1 = New-AzureRmNetworkInterface -ResourceGroupName $rgName -Location $location `

-Name 'Nic1' -LoadBalancerBackendAddressPool $bepool -NetworkSecurityGroup $nsg `

-LoadBalancerInboundNatRule $natrule1 -Subnet $vnet.Subnets[4]

$nicVM2 = New-AzureRmNetworkInterface -ResourceGroupName $rgName -Location $location `

-Name 'Nic2' -LoadBalancerBackendAddressPool $bepool -NetworkSecurityGroup $nsg `

-LoadBalancerInboundNatRule $natrule2 -Subnet $vnet.Subnets[4]

# Create an availability set.

$as = New-AzureRmAvailabilitySet -ResourceGroupName $rgName -Location $location `

-Name 'WEB-AVSet' -Sku Aligned -PlatformFaultDomainCount 3 -PlatformUpdateDomainCount 3

# Create two virtual machines.

# ############## VM1 ###############

# Create a virtual machine configuration

$vmConfig = New-AzureRmVMConfig -VMName 'WEB01' -VMSize Standard\_DS2\_v2 -AvailabilitySetId $as.Id | `

Set-AzureRmVMOperatingSystem -Windows -ComputerName 'WEB01' -Credential $cred | `

Set-AzureRmVMSourceImage -PublisherName MicrosoftWindowsServer -Offer WindowsServer `

-Skus 2016-Datacenter -Version latest | Add-AzureRmVMNetworkInterface -Id $nicVM1.Id

# Create a virtual machine

$vm1 = New-AzureRmVM -ResourceGroupName $rgName -Location $location -VM $vmConfig

# ############## VM2 ###############

# Create a virtual machine configuration

$vmConfig = New-AzureRmVMConfig -VMName 'WEB02' -VMSize Standard\_DS2\_v2 -AvailabilitySetId $as.Id | `

Set-AzureRmVMOperatingSystem -Windows -ComputerName 'WEB02' -Credential $cred | `

Set-AzureRmVMSourceImage -PublisherName MicrosoftWindowsServer -Offer WindowsServer `

-Skus 2016-Datacenter -Version latest | Add-AzureRmVMNetworkInterface -Id $nicVM2.Id

# Create a virtual machine

$vm2 = New-AzureRmVM -ResourceGroupName $rgName -Location $location -VM $vmConfig

# Install IIS on VM1 via CSE

#$PublicSettings = '{"commandToExecute":"powershell Add-WindowsFeature Web-Server"}'

#Set-AzureRmVMExtension -ExtensionName "IIS" -ResourceGroupName $rgName -VMName $vm1.Name `

# -Publisher "Microsoft.Compute" -ExtensionType "CustomScriptExtension" -TypeHandlerVersion 1.4 `

# -SettingString $PublicSettings -Location $location

# Install IIS on VM2 via DSC Push

#$PublicSettings = '{"ModulesURL":"https://github.com/Azure/azure-quickstart-templates/raw/master/dsc-extension-iis-server-windows-vm/ContosoWebsite.ps1.zip", "configurationFunction": "ContosoWebsite.ps1\\ContosoWebsite", "Properties": {"MachineName": "WEB02"} }'

#Set-AzureRmVMExtension -ExtensionName "DSC" -ResourceGroupName $rgName -VMName $vm2.Name `

# -Publisher "Microsoft.Powershell" -ExtensionType "DSC" -TypeHandlerVersion 2.19 `

# -SettingString $PublicSettings -Location $location

$End = Get-Date

Write-Host $start

Write-Host $end

\*\*There may be some errors while testing I received a long running VM error, the Azure VM agent was not installed and was not communicating to Azure Fabric.

New-AzureRmVM : Long running operation failed with status 'Failed'. Additional Info:'VM 'WEB02' has not reported status for VM agent or extensions. Please verify the VM has a running VM

agent, and can establish outbound connections to Azure storage.'

ErrorCode: VMAgentStatusCommunicationError

ErrorMessage: VM 'WEB02' has not reported status for VM agent or extensions. Please verify the VM has a running VM agent, and can establish outbound connections to Azure storage.

\*\* If you received an error like above, typically around an hour (timeout), then RDP into VM and check C:\ do you see a WindowsAzure folder ? If you do not the Windows\Linux Azure Agent was not installed and must be installed manually – [via Download and run Link](https://go.microsoft.com/fwlink/?LinkID=394789&clcid=0x409) | Once installed from Azure Portal Restart the VMs so the agent checks into the fabric.

Let’s check out the time, PowerShell will typically run synchronously single Azure resource item at a time, does not have redeploy, and would need additional PS logic for retry handling and dependency.

There are exceptions you can use [Start-Job -ScriptBlock {}](https://blogs.msdn.microsoft.com/mast/2016/06/29/microsoft-azure-how-to-execute-a-synchronous-azure-powershell-cmdlet-multiple-times-at-once-using-a-single-powershell-session/) with AzureRM and [Powershell Workflow](https://docs.microsoft.com/en-us/azure/automation/automation-powershell-workflow#parallel-processing) language to run cmdlets in parallel or asynchronously. Again you will have to write dependency and status checks around dependency.

Finally let’s explore Custom Script Extension (CSE) and Desired State Configuration (DSC) (Push method)

Custom Script Extension works as an imperative script launching after the OS communicates with the Azure Agent. It can be executed at time of creation in infrastructure ac code for PS,CLI,Terraform,ARM templates. It can also be launched anytime on running VM with the Azure Agent.

You can launch .ps1, .sh, and .py script files to execute.

You can even execute a CSE from the portal to a local script file.

Due to it’s imperative nature once the CSE runs successful it will never run again, so if IIS becomes uninstalled for example it won’t reinstall after a restart,

In the above PS there were remarked out ## Green areas, we can use these to run in PS and execute PS and DSC to install IIS on the VM OS.

Desired State Configuration is a declarative model where it will check if the conditions do not exist and then run, so if IIS is removed it will reinstall IIS for example.

Run the Following Commands in PS to deploy CSE and DSC:

# Install IIS on VM1 via CSE

$PublicSettings = '{"commandToExecute":"powershell Add-WindowsFeature Web-Server"}'

Set-AzureRmVMExtension -ExtensionName "IIS" -ResourceGroupName $rgName -VMName $vm1.Name `

-Publisher "Microsoft.Compute" -ExtensionType "CustomScriptExtension" -TypeHandlerVersion 1.4 `

-SettingString $PublicSettings -Location $location

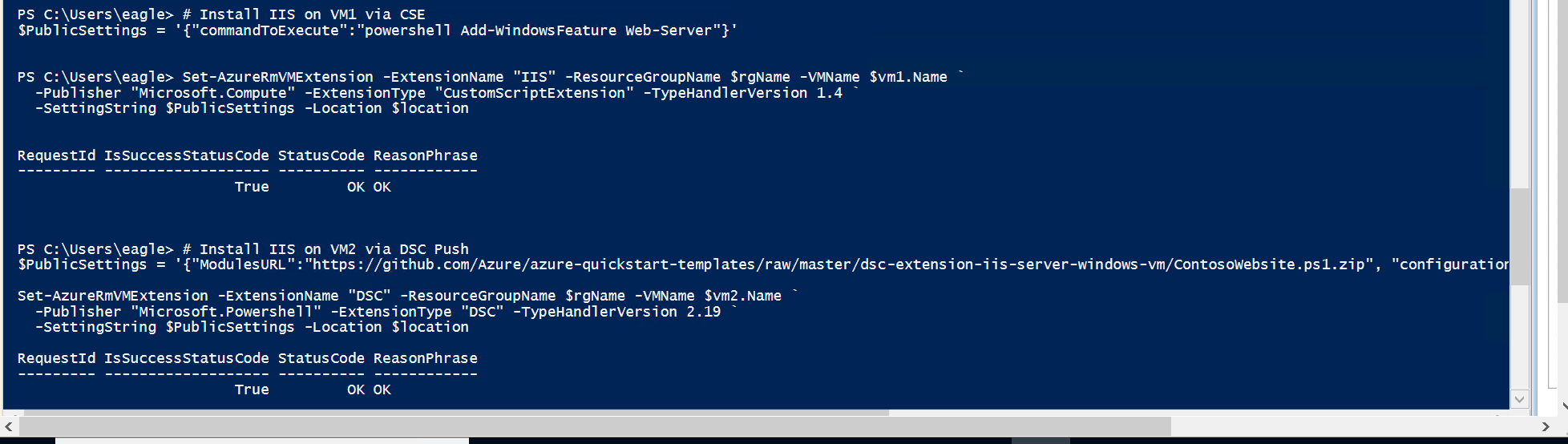
# Install IIS on VM2 via DSC Push

$PublicSettings = '{"ModulesURL":"https://github.com/Azure/azure-quickstart-templates/raw/master/dsc-extension-iis-server-windows-vm/ContosoWebsite.ps1.zip", "configurationFunction": "ContosoWebsite.ps1\\ContosoWebsite", "Properties": {"MachineName": "WEB02"} }'

Set-AzureRmVMExtension -ExtensionName "DSC" -ResourceGroupName $rgName -VMName $vm2.Name `

-Publisher "Microsoft.Powershell" -ExtensionType "DSC" -TypeHandlerVersion 2.19 `

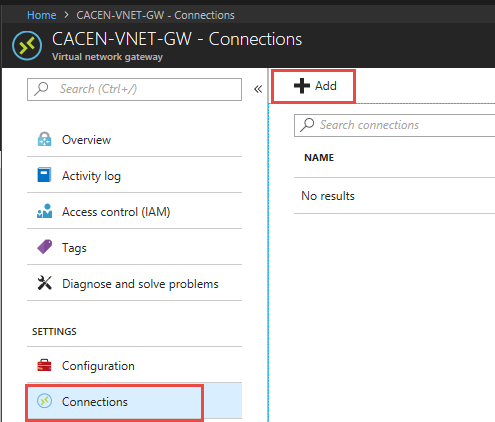
-SettingString $PublicSettings -Location $location



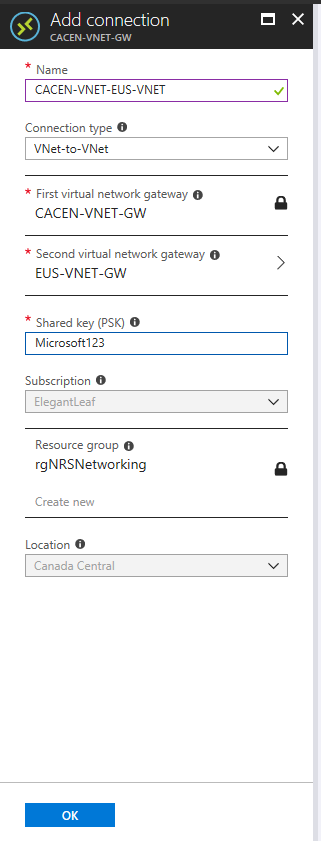
Once Completed, Examine the VM Extensions Blade

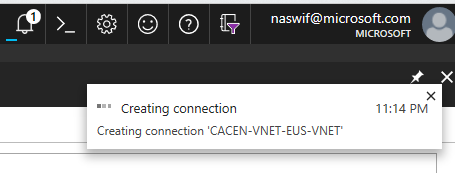
Let’s Start Connecting the Networks East US is not a location in Global VNET peering preview so we will create the traditional VNET-to-VNET connection over IPSEC S2S VPN to Canada Central.

Goto the virtual network gateway CACEN-VNET-GW, and goto the blade Connections and button + Add

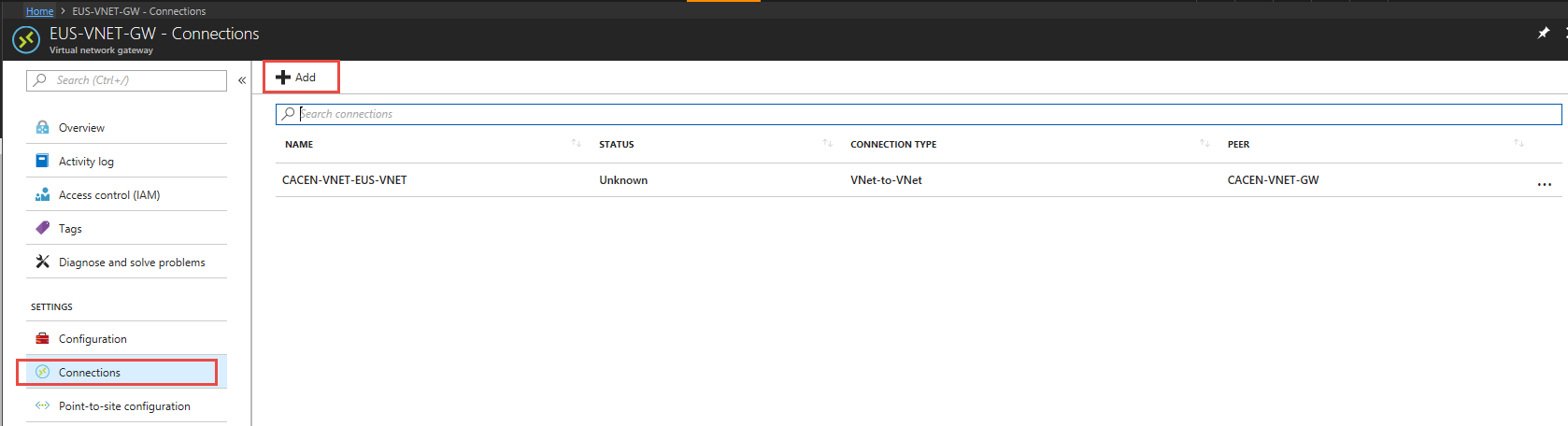


Fill out the connection configuration as follows

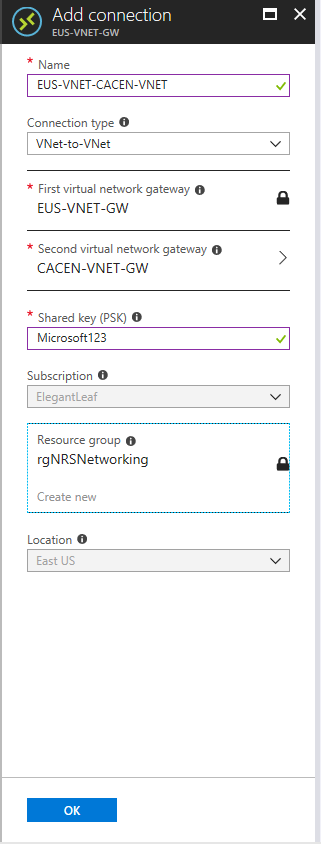




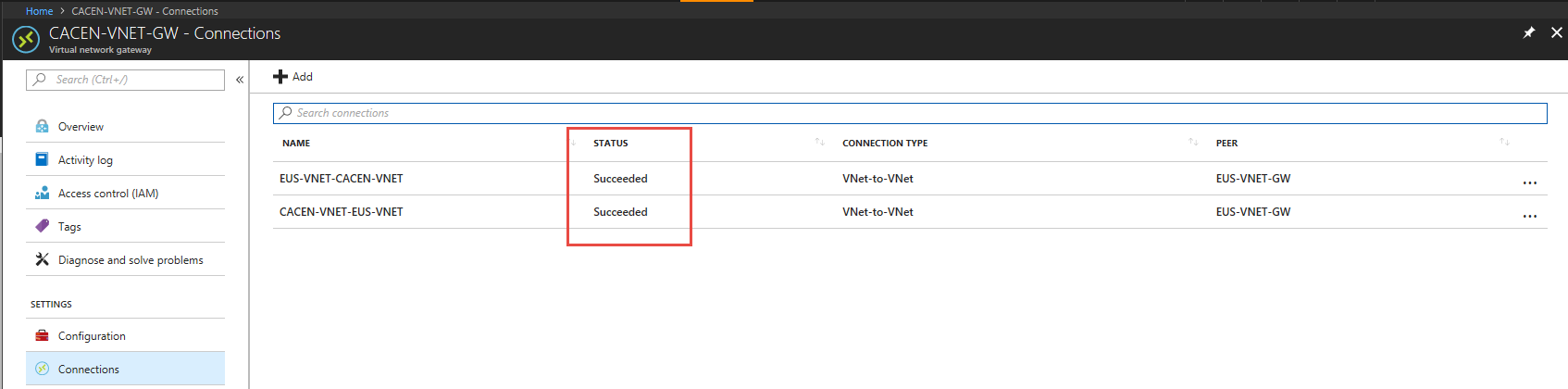
Goto the virtual network gateway EUS-VNET-GW, and goto the blade Connections and button + Add



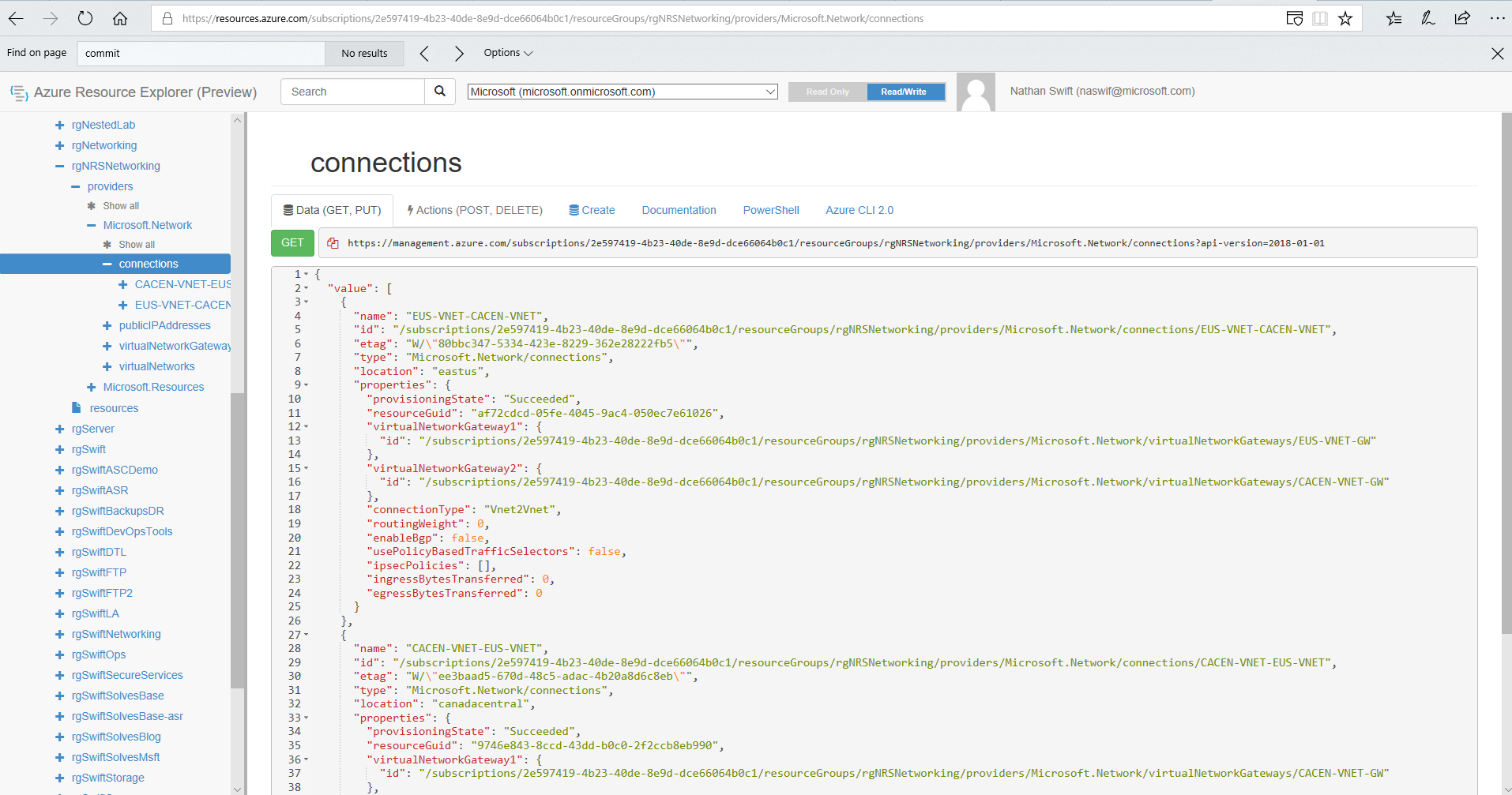
Fill out the connection configuration as follows



Eventually Status should show Succeeded, if not please delete the connections and recreate.



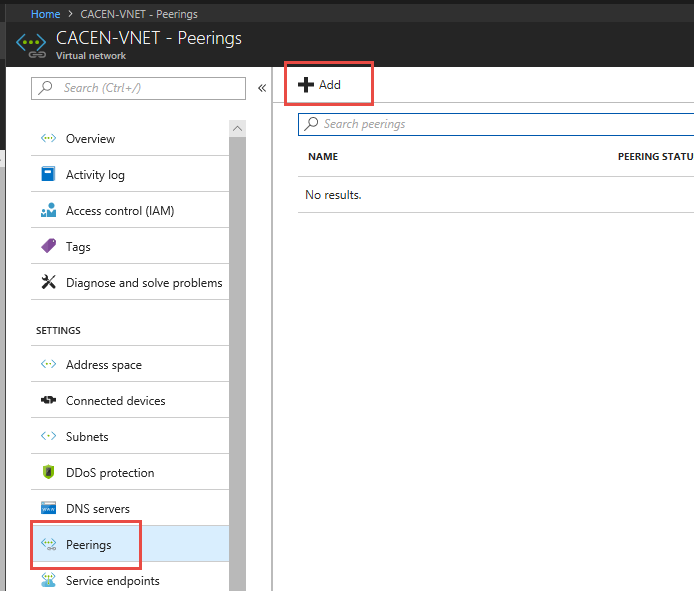
Check out the VNET-to-VNET connections Infrastructure as code in Azure Resource Manager, using the Resources Portal – <https://resources.azure.com>



This is a full management portal based on Azure Resource Manager in ARM JSON just like the ARM templates. This can be used to infer ARM template design. In addition you can use this portal to make configuration changes via REST API.

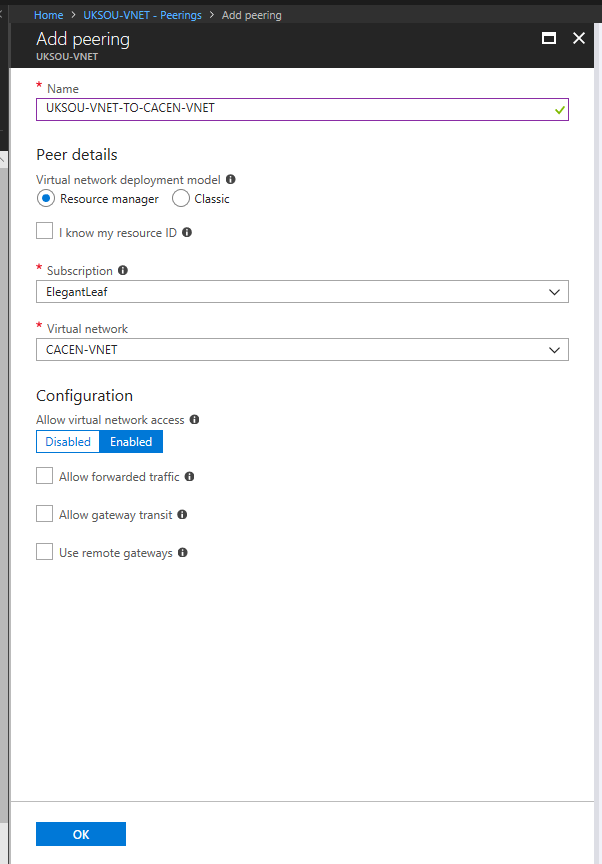
Canada Central and UK South are locations in preview that can be Global VNET paired

Goto the CACEN-VNET and goto the blade Peerings and + Add



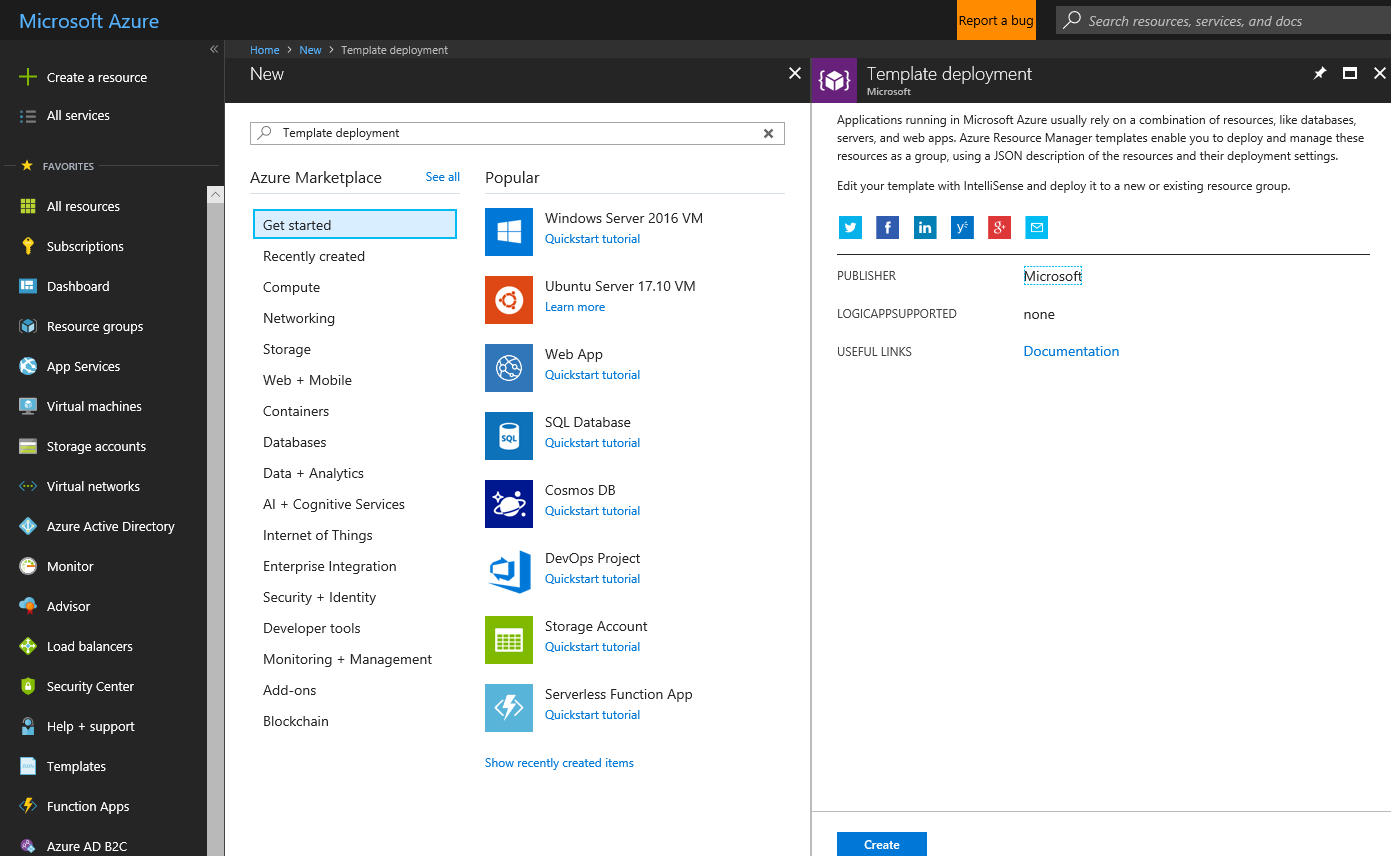


Goto the UKSOU-VNET and goto the blade Peerings and + Add

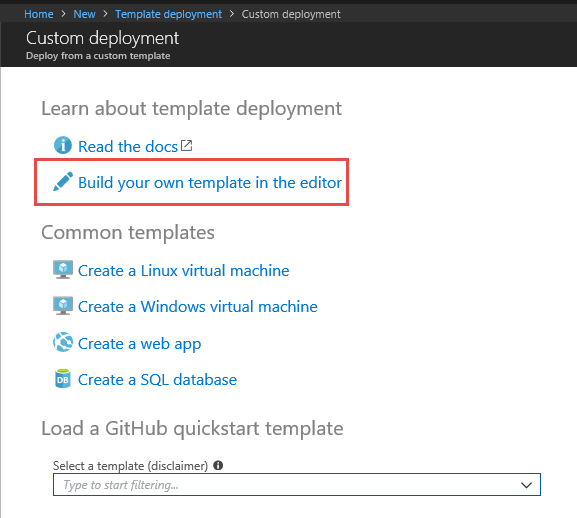


Now let’s deploy the same VM setup as we did in Azure PowerShell in Canada Central for East US using an ARM template instead.

Create a Resource + and search for Template Deployment



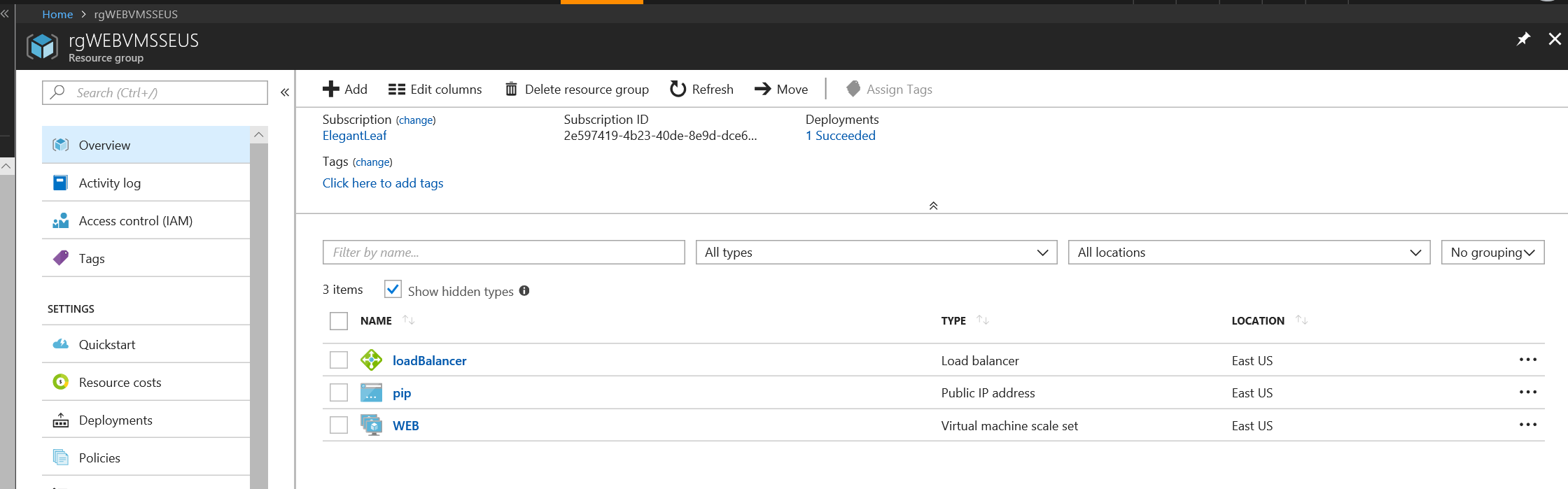
Build your own template in the editor



Clear out the 6 lines of skeletal structure and replace with the following and save:

<https://raw.githubusercontent.com/swiftsolves-msft/AzureIAC/master/scripts/WEBVMSSEUS.json>

Fill out the parameters and I accept terms and purchase



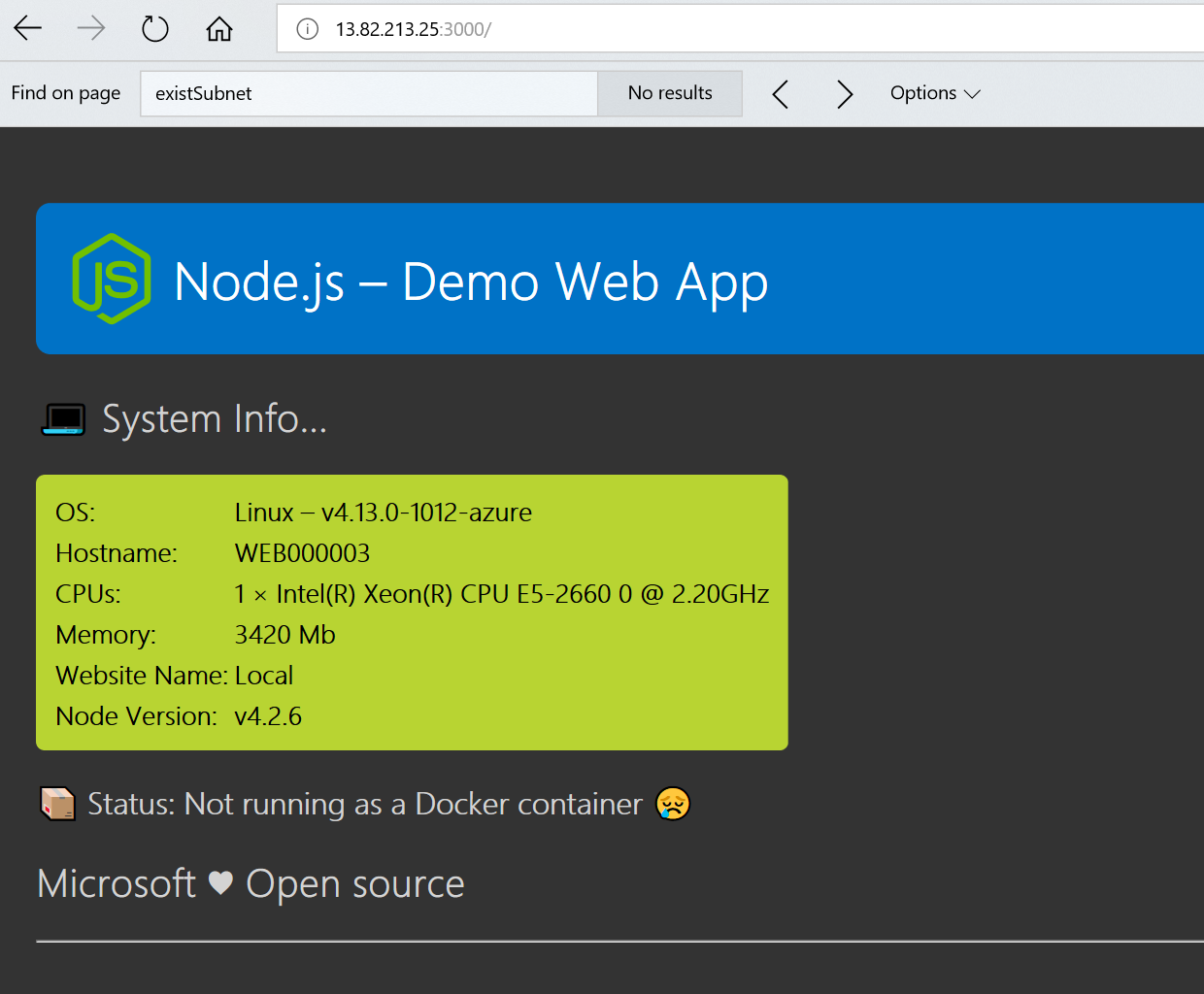
Lets check the LB Nat Rules and SSH into VMs in Scale Set

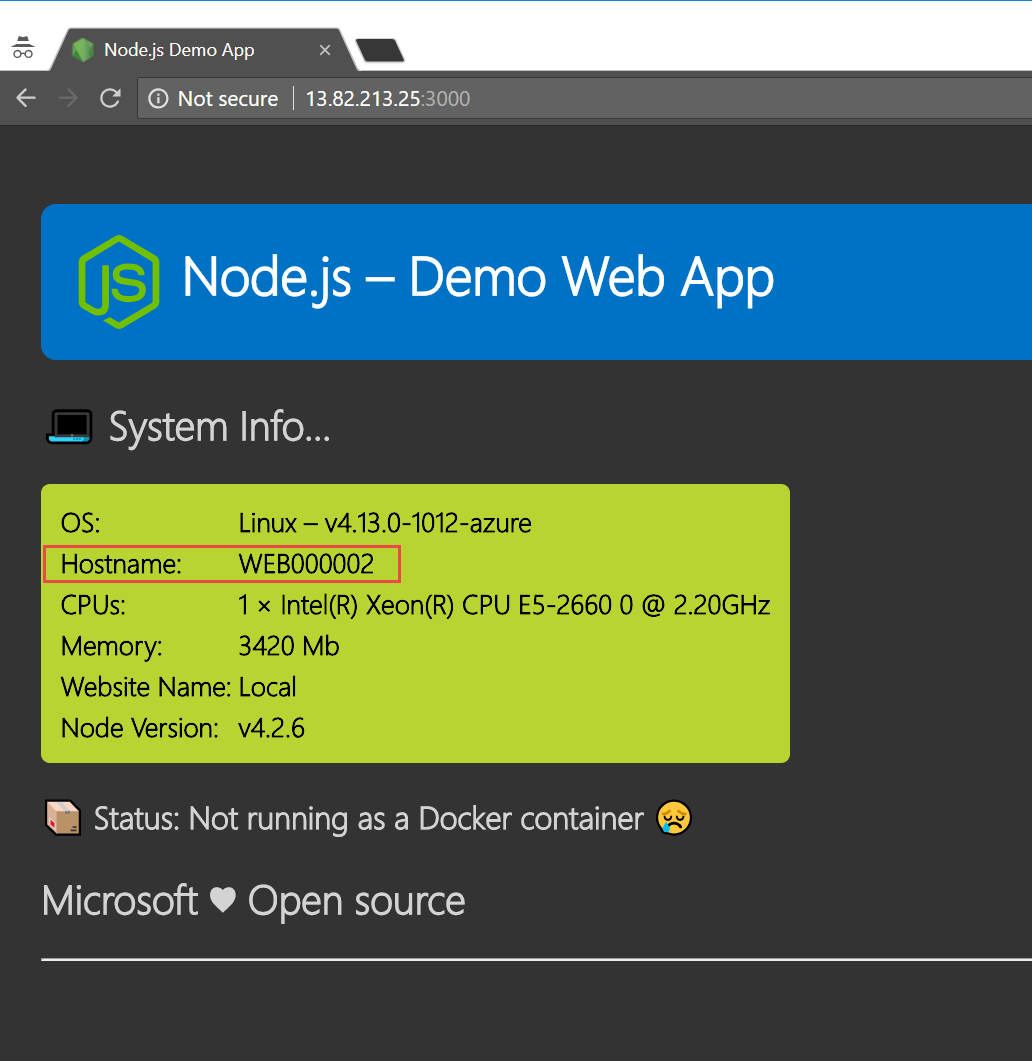
Ssh [xxxx@xx.xx.xx.xx](mailto:xxxx@xx.xx.xx.xx) -p 50000



Once SSH in go ahead and execute the script \ copy paste the script of: <https://github.com/swiftsolves-msft/AzureIAC/blob/master/scripts/DemoAppConfig.sh>

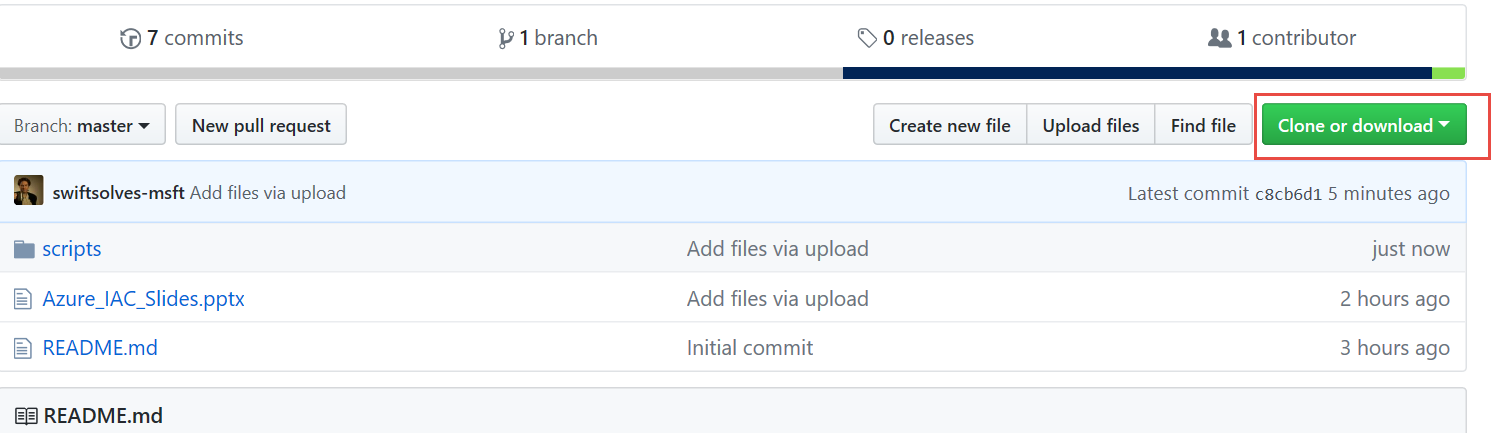
Web Browse to the Public IP Address of LB on :3000





Let use Terraform to deploy a couple VMs in UK South.

Download the following files from the Github scripts and place them into C:\Terraform

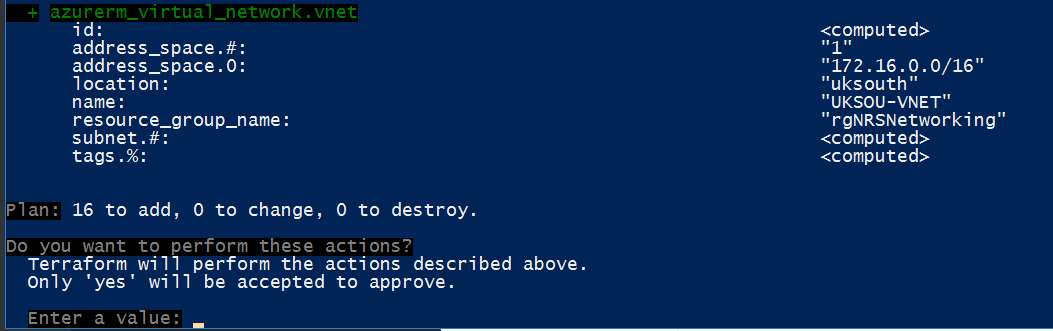


Main.tf

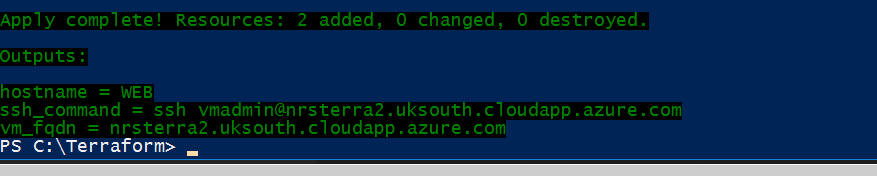
Variables.tf

Outputs.tf

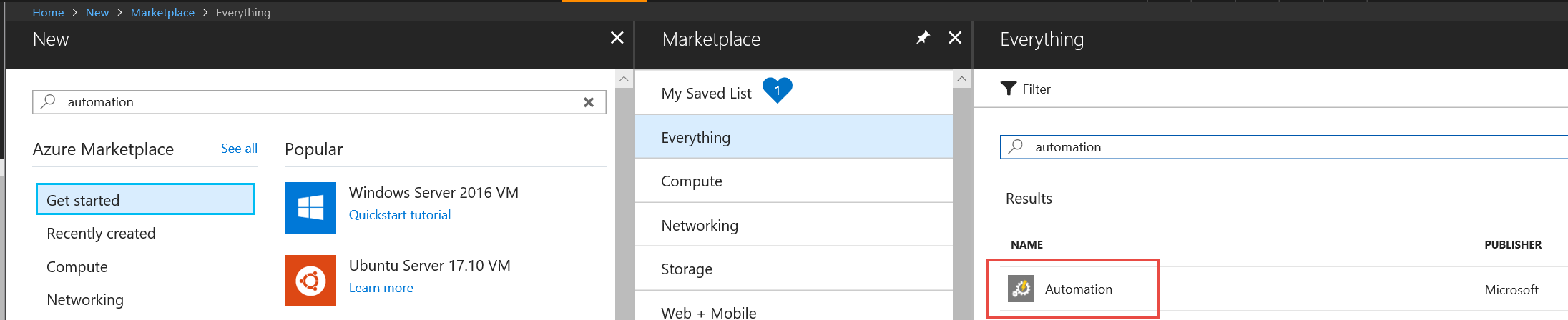
Run ./terraform.exe apply



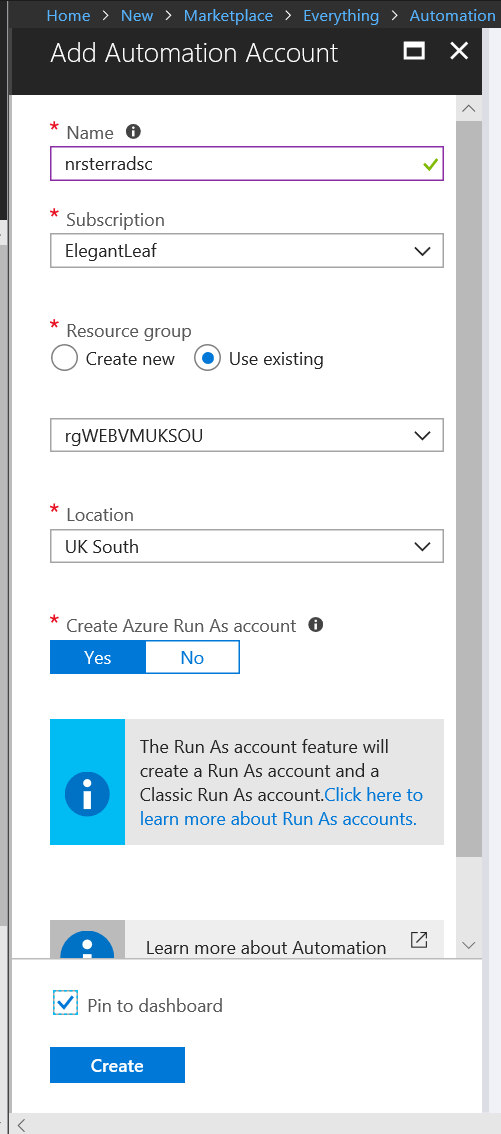
Type yes to confirm , Terraform is aware of resource dependency no need to call it out, Terraform executes in parallel.



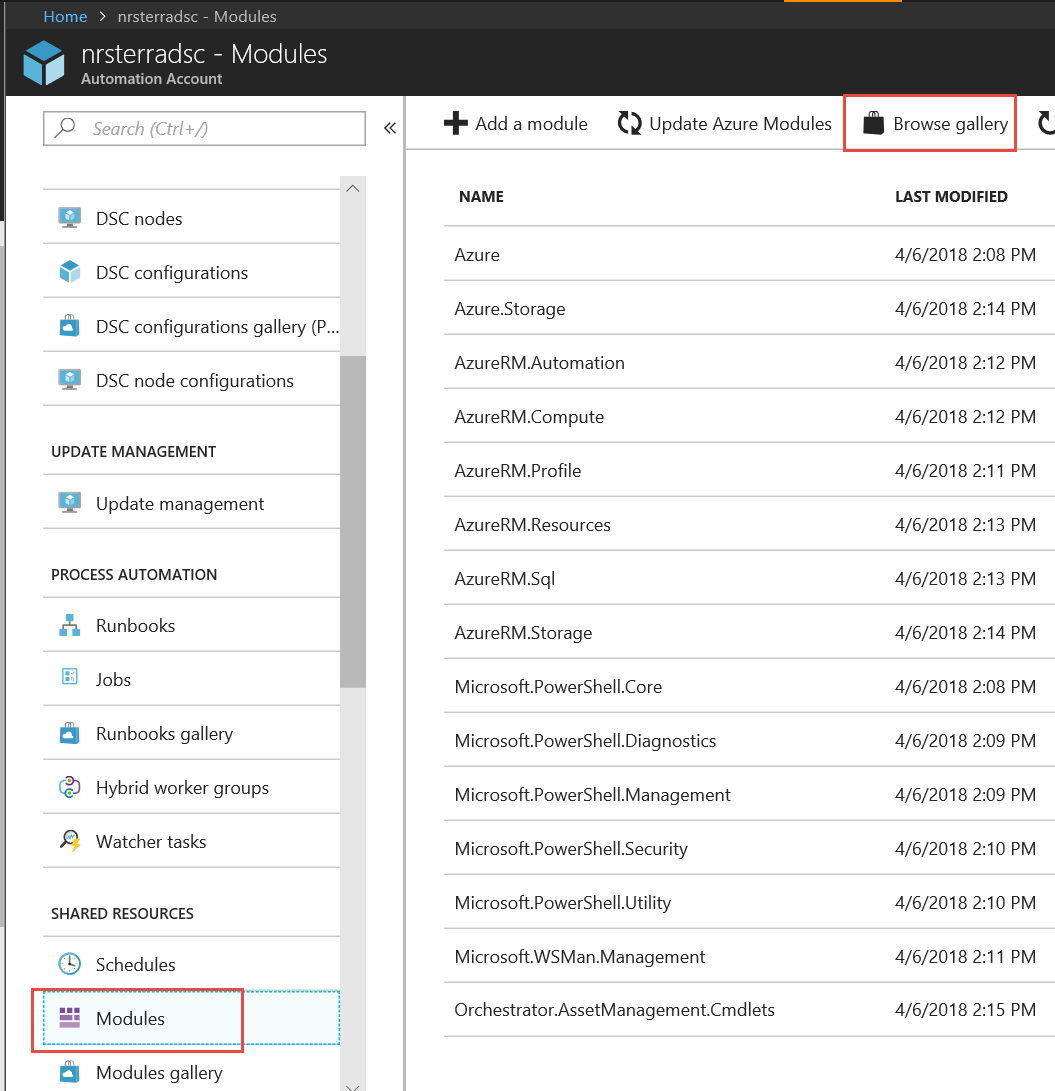
Let’s create a Azure Automation Account in UK South we will be using this for Desired State Configuration (DSC) (Pull)



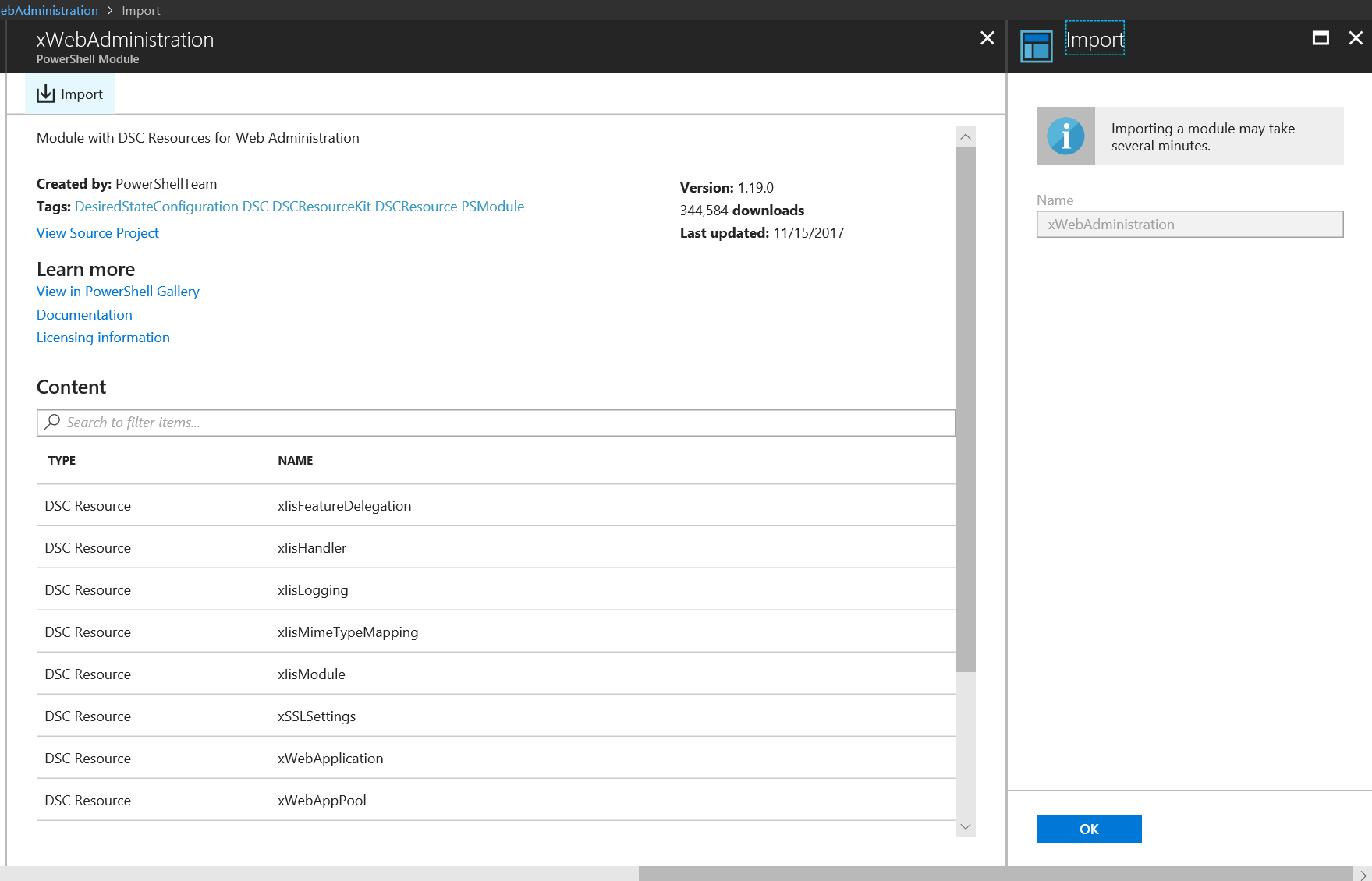
Ensure the name is unique the account name is part of a way to webhook and is also part of a fqdn url on internet

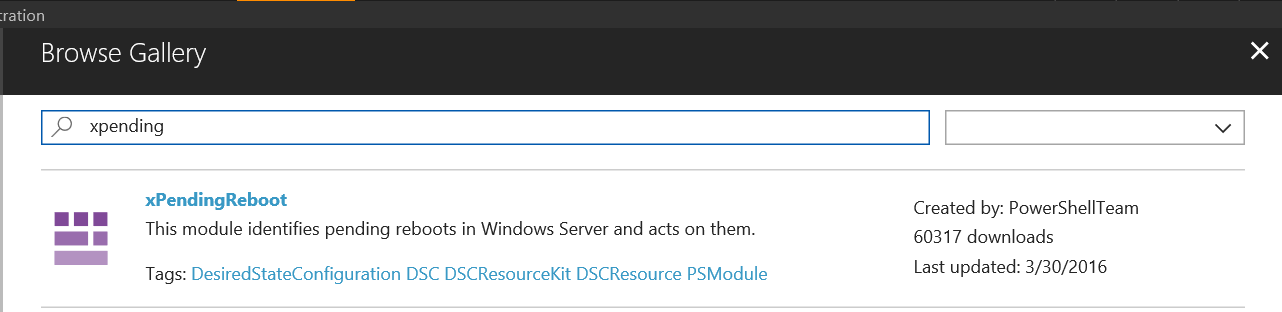


Once created we will import the following Modules for DSC – XwebAdministration and XPending Reboot



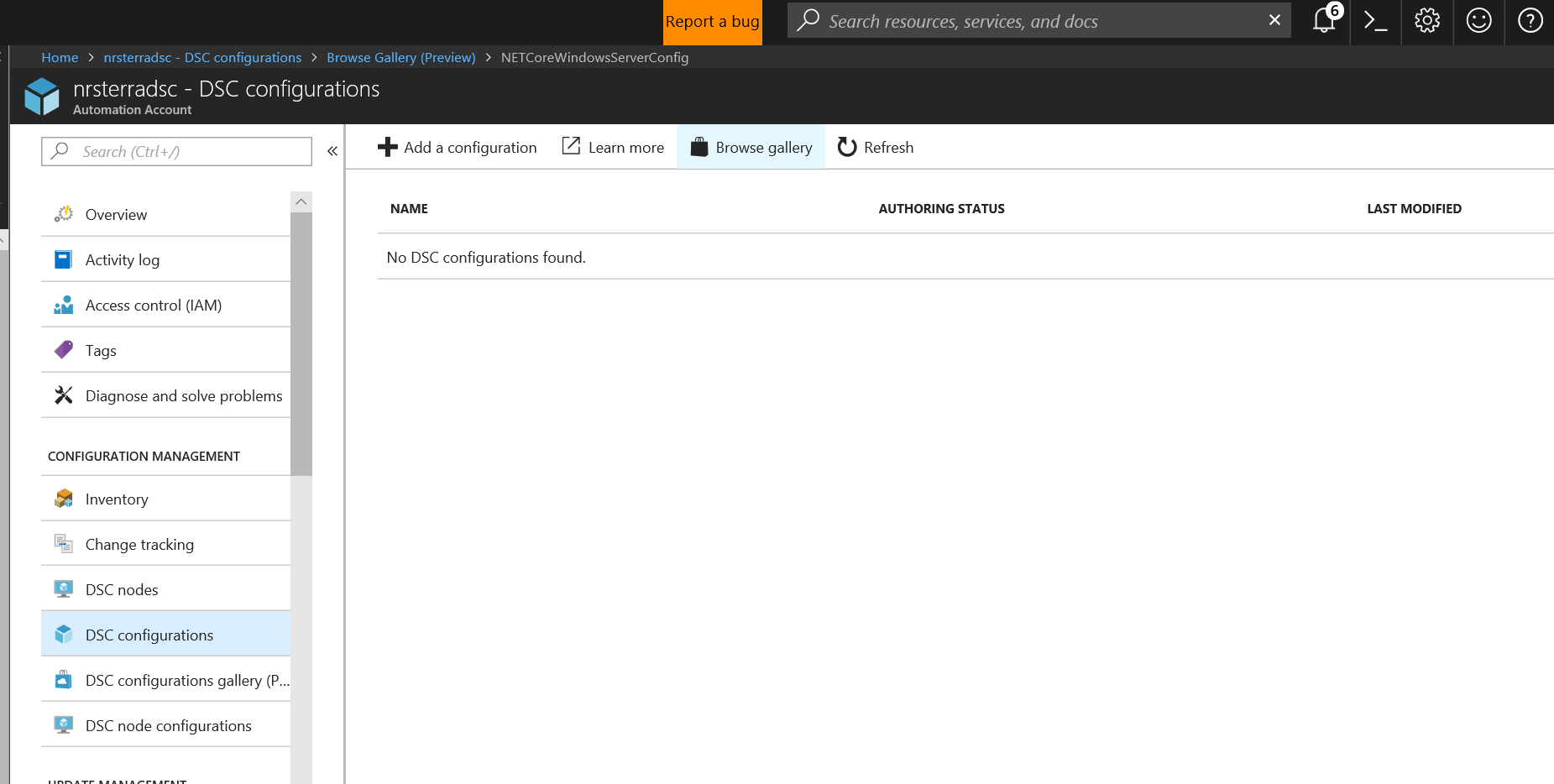


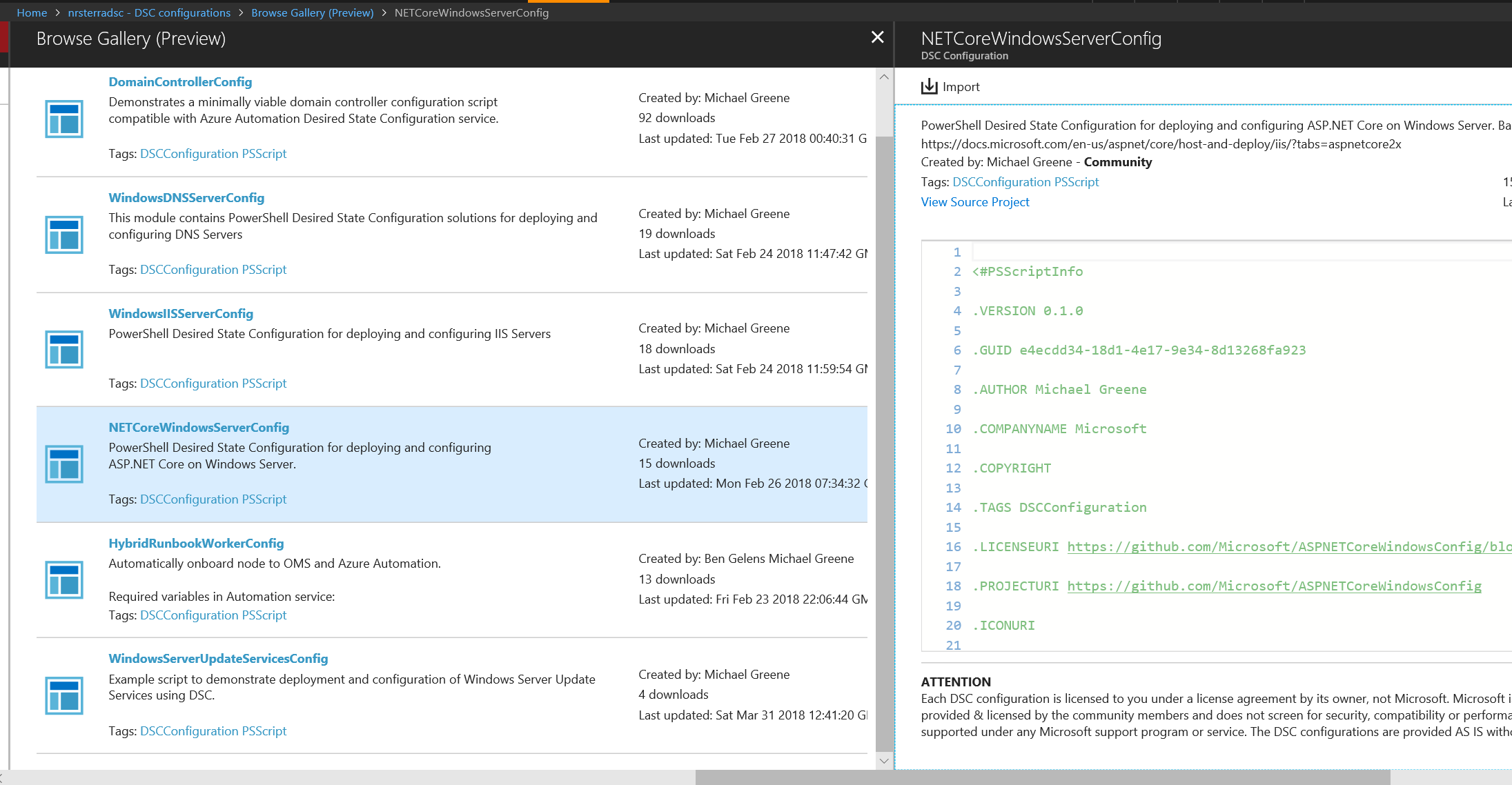




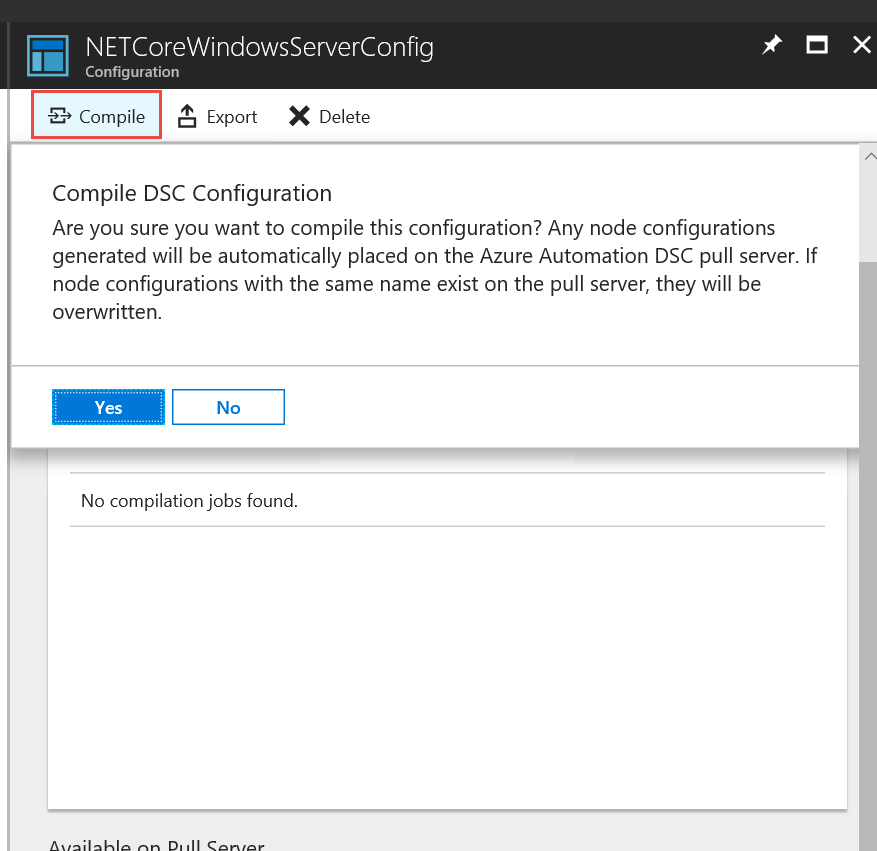
Once Modules succeed in Importing we will goto the DSC Configurations blade and import from gallery

***NETCoreWindowsServerConfig***

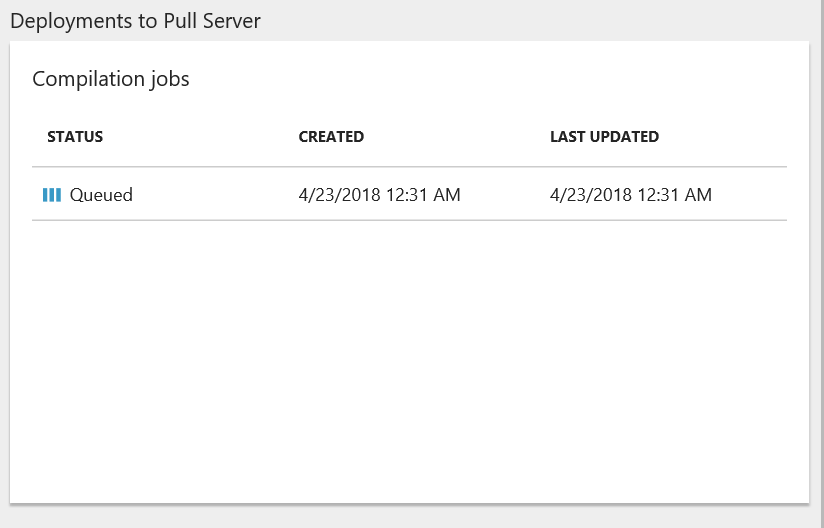


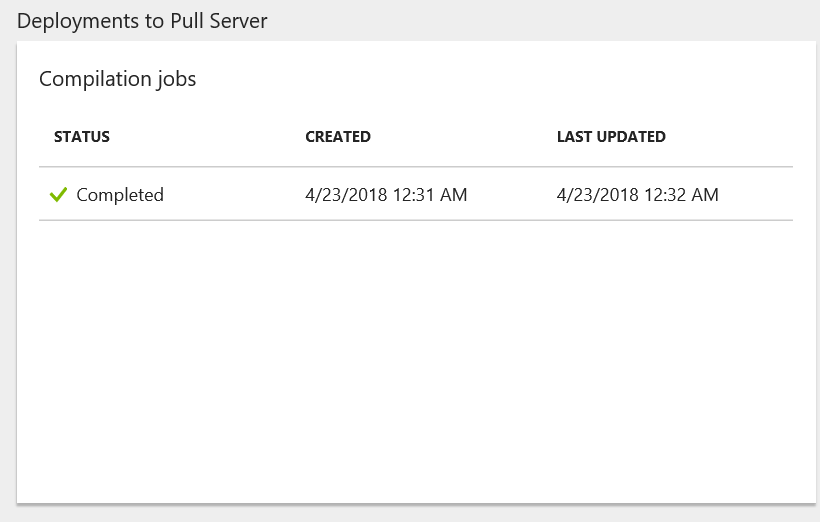


Next Compile the DSC Config to generate MOF files automatically

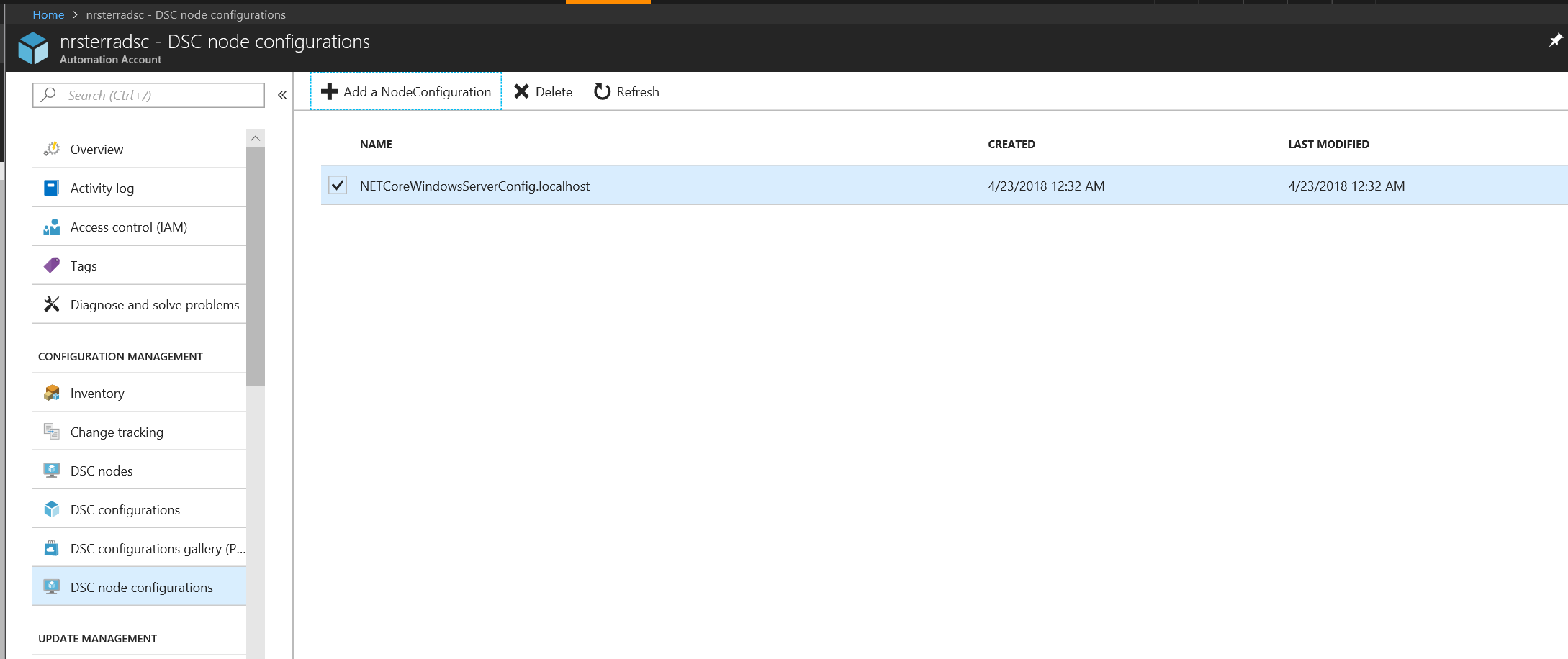


It will stay queued for a bit, wait until is is completed

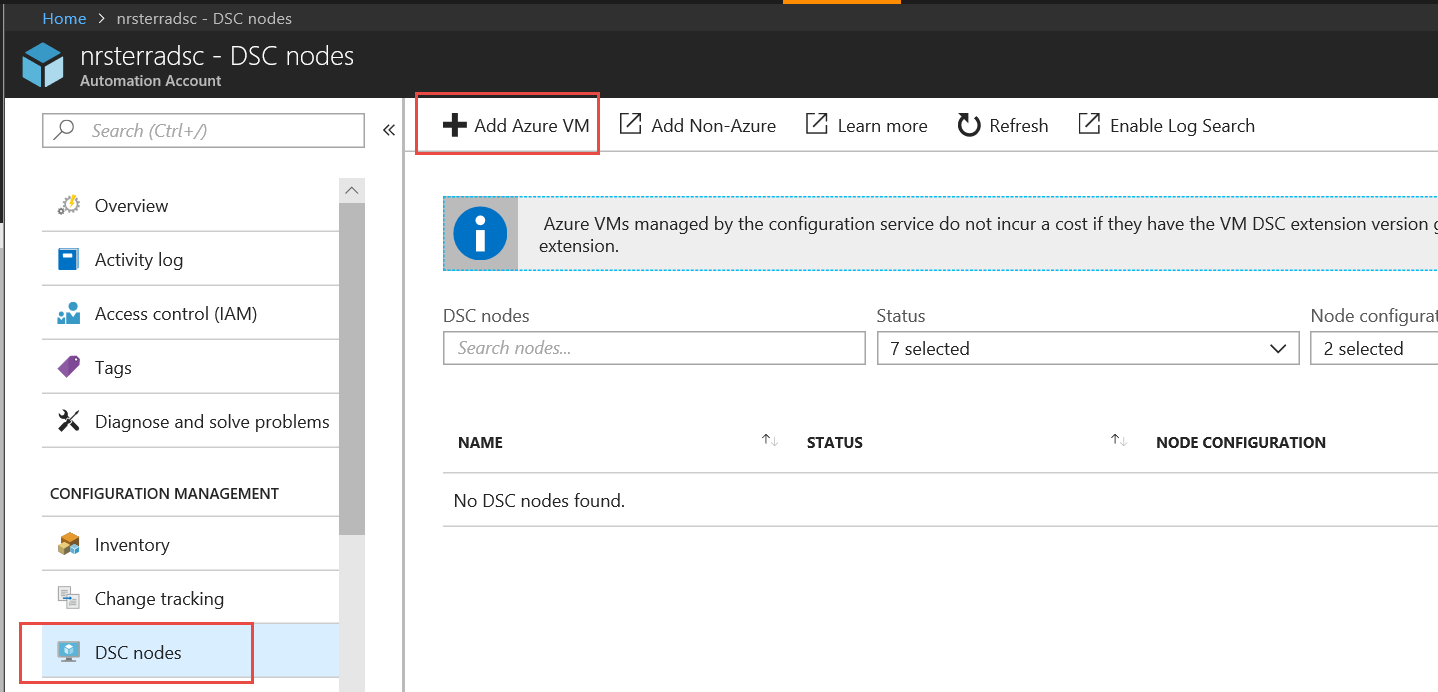


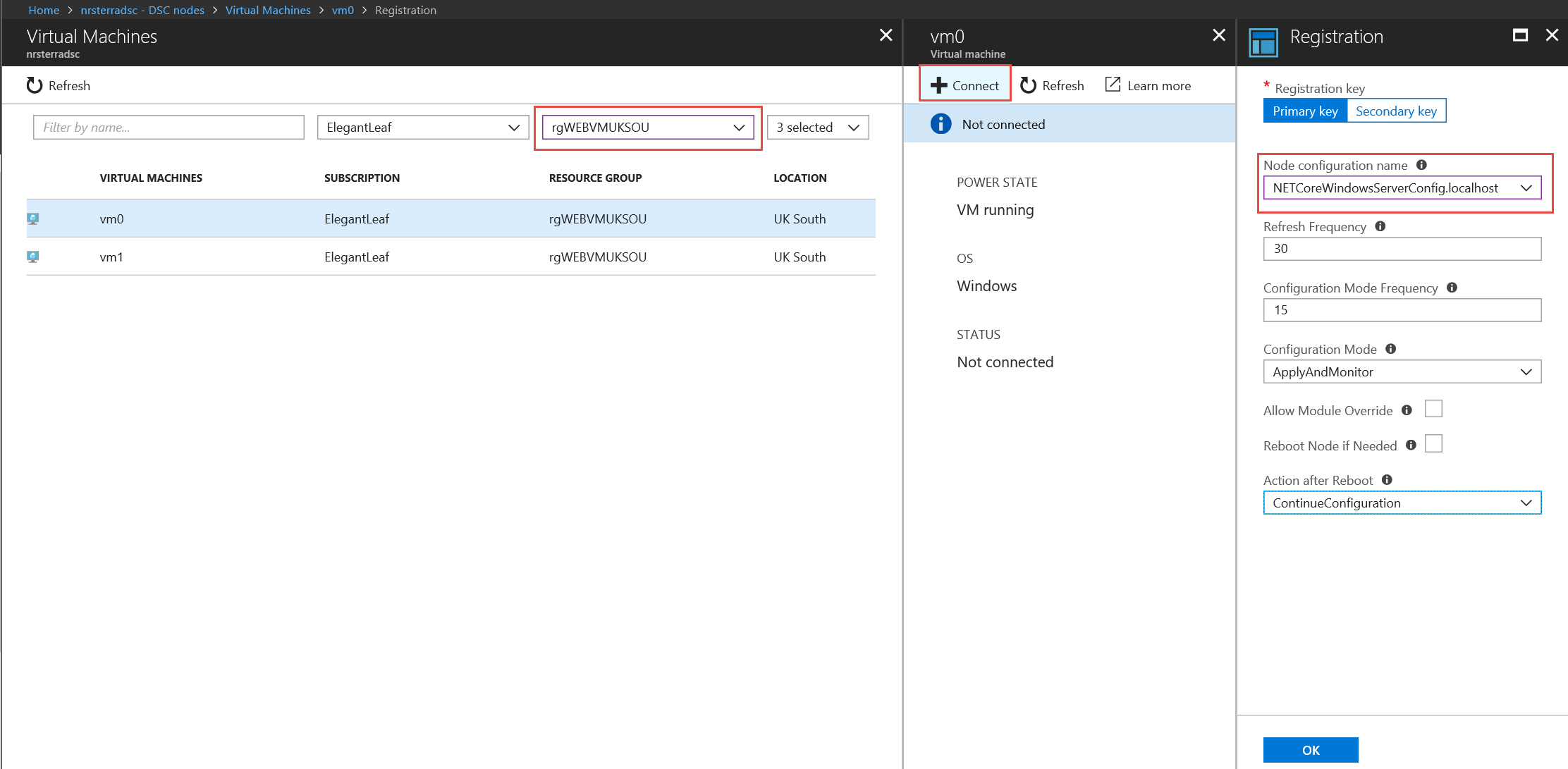


We now have a DSC Node Configuration for .NETCore



Goto DSC Nodes blade and + Add Azure VM





Add VM0 and VM1

**References:**

**ARM:**

[**Worldclass ARM Templates**](http://download.microsoft.com/download/8/E/1/8E1DBEFA-CECE-4DC9-A813-93520A5D7CFE/World%20Class%20ARM%20Templates%20-%20Considerations%20and%20Proven%20Practices.pdf)

[**ARM Template Reference**](https://docs.microsoft.com/en-us/azure/templates/microsoft.compute/virtualmachines)

[**Azure QuickStarts**](https://azure.microsoft.com/en-us/resources/templates/)

[**Ryan Jones Github**](https://github.com/rjmax?tab=repositories)

**Terraform:**

[**Terrafrom Github Examples**](https://github.com/terraform-providers/terraform-provider-azurerm/tree/master/examples)